POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Quantum Gradient Descent for Deep Learning Problems

SEYED MOJTABA MOHTASEBI
Département de Mathématiques et de génie industriel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Mathématiques appliquées

Mai 2025

© Seyed Mojtaba Mohtasebi, 2025.

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Quantum Gradient Descent for Deep Learning Problems

présenté par **Seyed Mojtaba MOHTASEBI**
en matrise *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Yann-Meing LAW-KAM-CIO, président
Steven DUFOUR, membre et directeur de recherche
Donatien N'DRI, membre

DEDICATION

To my beloved parents,

This work is dedicated to you as a testament to your unwavering love, endless support, and the sacrifices you have made throughout my life. Your encouragement, strength, and belief in me have been the foundation of every step I've taken in this journey. Without you, none of this would have been possible.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisor, Professor Steven Dufour, for his unwavering support, insightful guidance, and continuous encouragement throughout the course of this research. His expertise, patience, and constructive feedback have been instrumental in the development and completion of this thesis.

RÉSUMÉ

Cette thèse présente la conception et l'évaluation d'un réseau neuronal adaptatif quantique (QEANN-QGD), qui intègre des paradigmes d'apprentissage quantique et classique pour traiter efficacement du données complexes. Le modèle exploite la descente de gradient quantique (QGD) et intègre une Fusion de caractéristiques hybride (HFI), où les caractéristiques améliorées par le quantum sont combinées avec des caractéristiques extraites à l'aide de méthodes classiques. Cette architecture utilise les forces des deux paradigmes, garantissant une performance d'apprentissage évolutive et adaptative. Notre approche été évaluée sur des ensembles de données de référence, y compris MNIST et CIFAR-10, en utilisant diverses techniques d'optimisation telles que SGD et Adam comme points de référence. Pour l'ensemble de données MNIST on a atteint une précision de 99,26% avec QEANN-QGD, comparé à 99,84% avec SGD et 99,97% avec Adam, tout en maintenant une faible valeur de perte de 0,0281. Pour CIFAR-10, le modèle atteint une précision de 82,57%, restant compétitif a plusieurs modèles classiques, avec une valeur finale de perte de 0,2791. Les couches contribuent considérablement à capturer des motifs non triviaux, reflétés dans les courbes de précision et la convergence progressive de l'énergie de l'état fondamental au cours de l'optimisation. De plus, des techniques de modélisation et de réduction du bruit ont été appliquées pour atténuer les erreurs quantiques et assurer la fiabilité des calculs quantiques. En utilisant les circuits quantiques variationnels et les techniques de décalage de paramètres ont permis une optimisation efficace des caractéristiques quantiques. Les résultats montrent que notre entraînement basé sur la QGD offre une robustesse et une capacité d'extraction de caractéristiques comparables aux méthodes classiques conventionnelles. Cela démontre la capacité des systèmes hybrides quantiques-classiques à s'adapter à des ensembles de données diversifiés avec des améliorations en termes de précision, rappel, score F1 et ROC-AUC. Cependant, l'architecture hybride introduit des défis tels qu'une surcharge computationnelle accrue et des besoins en qubits, qui sont discutés en détail. Les travaux futurs se concentreront sur le développement de circuits quantiques plus efficaces pour réduire l'utilisation des qubits et explorer des algorithmes quantiques alternatifs pour l'optimisation afin d'améliorer encore les performances du modèle. Cette étude fait progresser l'apprentissage machine quantique en proposant un réseau neuronal hybride et évolutif qui équilibre avec succès les composant quantiques et classiques, avec des applications potentielles dans des domaines nécessitant une extraction de caractéristiques améliorées et des capacités d'apprentissage adaptatives.

ABSTRACT

This thesis presents the design and verification of a Quantum-Enhanced Adaptive Neural Network with quantum gradient descent (QEANN-QGD), which integrates quantum and classical learning paradigms to address complex data patterns effectively. The model leverages Quantum Gradient Descent(QGD) and incorporates Hybrid Feature Integration (HFI), where quantum-enhanced features are combined with classical features extracted through traditional methods. This architecture exploits the strengths of both paradigms, ensuring scalable and adaptive learning performance. The proposed approach is verified on benchmark datasets, including MNIST and CIFAR-10, using various optimization techniquessuch as SGD and Adam as baselines. For the MNIST dataset, we achieve an accuracy of 99.26% with QEANN-QGD, compared to 99.84% with SGD and 99.97% with Adam, while maintaining a low loss value of 0.0281. For CIFAR-10, the model has 82.57% accuracy, remaining competitive with several classical models, with a final loss value of 0.2791. The quantum layers significantly contributes to capturing non-trivial patterns, reflected in the gradual convergence of the ground-state energy during optimization. In addition, noise reduction techniques are applied to mitigate quantum errors and ensure the reliability of the quantum computations. Using variational quantum circuits and parameter-shift techniques enable effective optimization of quantum features. The results show that our QGD-based training provides comparable robustnessand feature extraction capabilities to conventional classical methods. But the classical counterpart showed better accuracy and precision when compared to hyprid model. The hybrid architecture introduces challenges such as increased computational overhead and qubit requirements, which are discussed in detail. Future work will focus on developing more efficient quantum circuit designs to reduce gthe number of qubit used and exploring alternative quantum algorithms for optimization to further enhance model performance. This study advances quantum machine learning by proposing a scalable, hybrid neural network that successfully balances quantum and classical components, with potential applications in domains requiring enhanced feature extraction and adaptive learning capabilities.

TABLE OF CONTENTS

DEDICATION iii

ACKNOWLEDGEMENTS iv

RÉSUMÉ v

ABSTRACT vi

TABLE OF CONTENTS vii

LIST OF TABLES ix

LIST OF FIGURES x

LIST OF SYMBOLS AND ACRONYMS xii

CHAPTER 1 INTRODUCTION 1
- 1.1 Motivation 1
- 1.2 Main Objectives 1
- 1.3 Thesis Outline 2

CHAPTER 2 LITERATURE REVIEW 3
- 2.1 Quantum mechanics 3
- 2.2 Computing foundations 3
- 2.3 Information theory 5
- 2.4 Quantum principal component analysis 5
  - 2.4.1 Process 6
- 2.5 Variational quantum eigensolver (VQE) in quantum machine Learning . . . 7
- 2.6 Quantum gates 11
- 2.7 Hybrid Quantum Autoencoders 14
- 2.8 Quantum multiclass classifier (QMCC) 15
- 2.9 Support Vector Machines With a Quantum Kernel Estimator (QSVM-Kernel) 17
- 2.10 Quantum fully self-supervised neural network (QFS-Net) 19
- 2.11 Quantum Deep Convolutional Neural Networks (QDCNN) 20
- 2.12 Long short-term memory neural networks 20
- 2.13 Recurrent quantum neuralnetworks (RQNN) 22

2.14 Quantum Recurrent Encoding–Decoding Neural Networks (QREDNN) . . . 23

CHAPTER 3 QUANTUM ENHANCED ADAPTIVE NEURAL NETWORK 26
3.1 Quantum Gradient Descent 26
3.1.1 Quantum Algorithm for Gradient Estimation 27
3.1.2 Quantum gradient descentalgorithm 31
3.2 Hybrid Feature Integration (HFI) 41
3.2.1 Integration with hybrid quantum-classical models 44
3.3 The adaptive method in quantum-classical neural networks 47
3.3.1 Parameter updateusing gradients 48
3.4 Noise implementation in quantum-enhanced adaptive neural networks (QEANN-QGD) 51
3.5 Noise reduction techniques in QEANN-QGD 52
3.6 Network architecture and implementation 56

CHAPTER 4 EXPERIMENTAL RESULTS 58
4.1 SGD performance on the MNIST dataset 58
4.2 Adam method performance on the MNIST dataset 60
4.3 SGD performance on the CIFAR 10 dataset 61
4.4 Adam method Performance on the CIFAR10 dataset 63
4.5 QEANN-QGD method results 65
4.6 QEANN-QGD method Performance on the mnist dataset 66
4.7 QEANN-QGD performance on the CIFAR10 dataset 68
4.8 Performance comparison across optimizers and datasets 70

CHAPTER 5 CONCLUSION 72
5.1 Summary of Works 72
5.2 Limitations 73
5.3 Future Research 74

REFERENCES 76

LIST OF TABLES

Table 4.1 Accuracy and Loss for Different Qubit Configurations 70
Table 4.2 Experimental results on MNIST and CIFAR-10 datasets 71

LIST OF FIGURES

Figure 2.1 Structure of the Hybrid Quantum Autoencoder, illustrating the encoding and decoding process with quantum circuits to compress data into a latent vector and reconstruct it 14

Figure 2.2 Diagram of the Quantum Variational Circuit model, showing state preparation, entanglement layers, and measurement on data and ancilla qubits 16

Figure 2.3 Quantum circuit structure for a support vector machine, demonstrating data encoding with Hadamard gates and unitary transformations for classification 17

Figure 2.4 QFS-Net architecture illustrating the fully self-supervised quantum neural network, with layers connected by qubit neurons for processing information in a multi-level quantum system 19

Figure 2.5 Structure of an LSTM cell, showing input, forget, and output gates that regulate information flow for long short-term memory processing in sequential data 21

Figure 2.6 Schematic representation of Quantum Neural Network (QNN) and Recurrent Quantum Neural Network (RQNN) models, illustrating the flow of information and parameter updates in both architectures 23

Figure 2.7 Structure of the Quantum Recurrent Encoder-Decoder Neural Network (QREDNN), highlighting the attention mechanism, encoder, and decoder components for sequential data processing 24

Figure 3.1 Schematic of the gradient calculation process, illustrating steps from the initial state prepared by CNN to the optimization of the loss function 26

Figure 3.2 Schematic representation of quantum gradient descent, showing operations from initial state preparation with H and R(z) gates to measurement through QSUB, QADD, and IQFT step 27

Figure 3.3 Quantum circuit illustrating the implementation of the Quantum Gradient Descent algorithm, showing the arrangement of quantum gates for parameter optimization 34

Figure 3.4 3D plots of the loss function trajectory starting from points (0.5, 1.5) and (1.2, -1.8), illustrating the optimization path taken 39

Figure 3.5 3D plots of the loss function trajectory starting from points (-1.8, 1.0) and (-1.5, -1.5), illustrating the optimization path 40

Figure 3.6 Quantum-Enhanced Artificial Neural Network with Quantum Gradient Descent (QEANN-QGD) diagram illustrating the network architecture, including input, convolutional layers, maxpooling, quantum layer, and fully connected layer stages 44
Figure 4.1 Stochastic Gradient Descent(SGD) accuracy over MNIST dataset, showing the progression of accuracy across 25 epochs 58
Figure 4.2 Stochastic Gradient Descent (SGD) loss over MNIST dataset, illustrating the decrease in loss across 25 epochs 59
Figure 4.3 Accuracy of Adam optimizer over MNIST dataset, showing the trend of accuracy across 25 epochs 60
Figure 4.4 Loss of Adam optimizer over MNIST dataset, depicting the reduction in loss across 25 epochs 61
Figure 4.5 Stochastic Gradient Descent (SGD) accuracy over CIFAR-10 dataset, showing the increase in accuracy across 25 epochs 62
Figure 4.6 Stochastic Gradient Descent (SGD) loss over CIFAR-10 dataset, illustrating the decrease in loss across 25 epochs 63
Figure 4.7 Accuracy of Adam optimizer over CIFAR-10 dataset, showing the progression of accuracy across 25 epochs 64
Figure 4.8 Loss of Adam optimizer over CIFAR-10 dataset, illustrating the reduction in loss across 25 epochs 65
Figure 4.9 Performance of Quantum-Enhanced Artificial Neural Network with Quantum Gradient Descent (QEANN-QGD) over the MNIST dataset, showing model accuracy and loss across 25 epochs 66
Figure 4.10 Convergence of Ground State Energy for the MNIST dataset using QEANN-QGD, comparing the model's energy values with the theoretical ground state energy across 25 epochs 67
Figure 4.11 Performance of Quantum-Enhanced Artificial Neural Network with Quantum Gradient Descent (QEANN-QGD) over CIFAR-10 dataset, displaying accuracy and loss trends across 100 epochs 68
Figure 4.12 Convergence of Ground State Energy for the CIFAR-10 dataset using QEANN-QGD, comparing the model's energy values with the theoretical ground state energy across 100 epochs 69

LIST OF SYMBOLS AND ACRONYMS

CNN	Convolutional NeuralNetwork
EEG	Electroencephalogram
FFNN	Feed Forward Neural Networks
HFI	Hybrid Feature Integration
HKNN	Hybrid k-neighbors-nearby model
HQA	Hybrid Quantum Autoencoders
IQFT	Inverse Quantum Fourier Transform
IQP	Instantaneous Quantum Polynomial time
LHC	Large Hadron Collider
LSTM	Long Short-Term Memory
NISQ	Noisy Intermediate-Scale Quantum
NMSE	Normalized Mean-Square Errors
PQC	Parameterized Quantum Circuit
QADD	Quantum Adder
QANN	Quantum Artificial Neural Networks
QBP	Quantum Backpropagation Neural Networks
QDCNN	Quantum Deep Convolution Neural Networks
QFS-Net	Quantum Fully Self-Supervised Neural Network
QFT	Quantum Fourier Transform
QGAN	Quantum Generative Adversarial Networks
QGD	Quantum Gradient Descent
QGRU	Quantum Gated Recurrent Units
QMCC	Quantum Multiclass Classifier
QML	Quantum Machine Learning
QNN	Quantum Neural Network
QPCA	Quantum Principal Component Analysis
QRC	Quantum Reservoir Computing
QREDNN	Quantum Recurrent Encoding–Decoding Neural Networks
QRL	Quantum Reinforcement Learning
QSVM	Quantum Support Vector Machine
QSUB	Quantum Subtractor
ReLU	Rectified Linear Unit
ROC	Receiver Operating Characteristic

RX	Rotation-X Gate
RZ	Rotation-Z Gate
RQNN	Recurrent Quantum Neural Networks
SGD	Stochastic Gradient Descent
VQC	Variable Quantum Circuits
VQE	Variational Quantum Eigensolver

CHAPTER 1 INTRODUCTION

1.1 Motivation

The rapid advancements in quantum computing have opened new possibilities for studying complex problems, including in the domain of machine learning. Classical machine learning techniques, while powerful, often face challenges when dealing with high-dimensional data, complex patterns, or computational overhead, especially as dataset sizes grow exponentially. With the advent of quantum machine learning (QML) [1, 2], researchers are exploring hybrid approaches that combine the strengths of classical models with the power of quantum computation to unlock higher efficiency and enhanced predictive performance. A particularly promising application is the integration of quantum-enhanced algorithms such as the quantum gradient descent (QGD) method [2] into existing machine learning frameworks. This opens the door for models capable of handling more complex data representations faster, such as image classification tasks on datasets like MNIST and CIFAR 10.

However, the practical realization of quantum algorithms in neural networks remains an ongoing challenge. Noise, scalability, and hardware limitations are still areas of active research, which motivates the investigation of hybrid methods. In this context, this thesis proposes a quantum-enhanced adaptive neural network with quantum gradient descent (QEANN-QGD) framework that leverages quantum gradient descent to increasethe performance of classical architectures. This research shows the potential of combining quantum circuits with Convolutional Neural Networks (CNNs) and fully connected layers to tackle high-dimensional data efficiently.

1.2 Main Objectives

The specific objectives of this thesis are :

1. **Designe a hybrid quantum-classical architecture :** Develop a neural network model that integrates quantum circuits with classical layers for feature extraction and data processing ;
2. **Implement quantum gradient descent (QGD) method :** Incorporate QGD as an optimization technique within the hybrid architecture, leveraging quantum computing's strengths for faster convergence and better model performance ;
3. **Compare the performance of methods :** evaluate the performance of the proposed hybrid model against traditional optimization techniques such as the stochastic

gradient descent method and Adam method, particularly in tasks involving image classification ;

4. **Explore scalability and practicality :** Examine the challenges of scaling the hybrid quantum-classical model to larger datasets MNIST and CIFAR 10 , and real-world applications, with a focus on the limitations and benefits of quantum hardware.

1.3 Thesis Outline

The structure of this thesis goes as follows :

- **Chapter 1 : Introduction :** Provides an overview of the context in which quantum computing intersects with machine learning, along with a detailed explanation of the motivation behind hybrid models ;
- **Chapter 2 : Literature review :** Reviews existing work in quantum machine learning, neural networks, and quantum optimization techniques, positioning this research within the broader academic discourse ;
- **Chapter 3 : QEANN-QGD :** Describes the architecture of the **QEANN** model, including the quantum feature extraction process, classical CNN layers, and the quantum gradient descent optimization method ;
- **Chapter 4 : Experimental results :** Assess the results of applying the hybrid model to image classification tasks, analyzing its performance compared to classical models;
- **Chapter 5 : Conclusion and future work :** Summarizes the findings of the research and discusses potential directions for further exploration, including the application of the model to more complex datasets and its integration with quantum hardware platforms.

CHAPTER 2 LITERATURE REVIEW

2.1 Quantum mechanics

In the early 20th century, physics faced new challenges. Experiments showed that classical mechanics had some major gaps. This led to the development of quantum mechanics, a new way of understanding the nature. Werner Heisenberg began the development quantum mechanics in 1925 [3]. Quantum mechanics is a set of mathematical rules that form the basis for understanding this theories [4].

Quantum computing is one the fields that originated from quantum mechanics. Richard Feynman started the idea that a computer could work based on quantum mechanics principles [5].

Quantum mechanics and quantum information theory are important components of exploring quantum computing. They challenge researchers to develop new methods for controlling individual quantum systems, leading to innovative experiments and guiding future research. Gaining precise control over these systems is essential to harnessing the full potential of quantum mechanics for practical applications in quantum computing and information processing [6].

Despite growing interest, progress in developing quantum information processing systems has been modest. Today's most powerful quantum computers can perform only a limited number of operations with a small number of qubits.

The twomost important ideas of quantum mechanics used in quantum computing are superposition and entanglement [7]. Superposition means a system can exist in multiple states at once until measured. A qubit, for instance, can simultaneously be in "0" and "1" states. This allows quantum computers to process large amounts of information in parallel. Entanglement is a phenomenon where two or more quantum particles becomecorrelated in such a way that the state of one particle instantly influences the state of the other, regardless of distance. This enables quantum computers to perform complex computations more efficiently, as entangled qubits share information instantaneously.

2.2 Computing foundations

In 1936, mathematician Alan Turing introduced the concept of a programmable computer, known as the Turingmachine [8]. He showed that a Universal Turing Machine could perform

any calculation that any other computer could, supporting the idea that any algorithm could be run on this model. This concept, later called the Church-Turing thesis (developed with Alonzo Church), connects real-world algorithms to their theoretical equivalents [9]. This idea became widely accepted and is the foundation of modern computer science [10].

In 1985, David Deutsch proposed a stronger version of the Church-Turing thesis based on physical laws [11]. He suggested that a device could efficiently model any physical systems, using quantum mechanics as the foundation. This idea led to the concept of quantumcomputers, which are quantum versions of Turing machines, that laid the groundwork for modern quantum computing.

In 1994, Shor built on Deutsch's work by showing that quantum computers could efficiently solve problems that were considered intractable for classical computers, such as large-integer factoring and the discrete logarithm problem. This breakthrough gained widespread attention and reinforced the belief that quantum computers possess computational abilities that surpass those of classical systems. Notably, this includes outperforming even classical probabilistic algorithms, which use randomness to explore solution spaces. While probabilistic algorithms can offer speed-ups for certain problems, they still operate within the bounds of classical computation. In contrast, Shor's algorithm demonstrated that quantum entanglement and superposition enable fundamentally different approaches to computation, allowing exponential speed-up for problems like factoring a feat believed to be unattainable even for randomized classical methods.

More evidence for quantum computers' advantages came in 1996 when Lov Grover introduced an algorithm that sped up searches in unstructured data sets [12]. While not as fast as Shor's algorithm, Grover's algorithm was important because it applied to many search problems, such as searching for a marked item in an unsorted database, solving instances cryptographic functions. Theseexamples demonstrate how Grover's approach can enhance a broad class of computational tasks, showing another clear benefit of quantum computing.

At the same time as Shor's and Grover's algorithms were developed, researchers explored Richard Feynman idea in 1982. Feynman stated that classical computers struggle to simulate quantum systems and that quantum computers might overcome this.

Comparing quantum and classical computers raises important questions about their strengths and weaknesses. What makes quantum computers better for certain tasks ? And which problems can they solve that classical computers cannot ? The answers are still unclear and will require much more research to fully understand the potential of quantum computing.

2.3 Information theory

As quantum computation progress, it's important to consider another key area in its development : information theory. In 1948, Claude Shannon published foundational papers that created modern information and communication theory [13]. His work deeply impacted data encoding and transmission, influencing fields like quantum computing.

Quantum information theory has developed alongside classical information theory. In 1995, Ben Schumacher adapted Shannon's noiseless coding theorem to quantum mechanics, introducing the "quantum bit", or "qubit", as a resource [14]. While there is no quantum version of Shannon's noisy-channel theorem yet, a theory of quantum error correction has been developed. This allows quantum computers to function reliably in noisy environments, similar to error correction in classical information theory.

Networked quantum information theory is still in its early stages. A major question is how much information can be sent through quantum channel networks. While recent experiments are promising, a unified theory for networked quantum channels is still under development.

Quantum computing has made significant progress in improving algorithmic efficiency, especially for searchand counting problems. An important advancement is Quantum Amplitude Amplification and Estimation, introduced by Brassard, Høyer, Mosca, and Tapp [15]. This technique builds on Grover's algorithm [12], extending its use and optimizing performance for a wider range of computational problems.

Grover's original quantum search algorithm was a key breakthrough in quantum computing, showing a quadratic speedup over classical methods for searching an unsorted database. It increased the probability of finding a target element with each iteration. However, the algorithm had limitations, such as requiring a known number of solutions and assuming each element in the search space had an equal chance of being the target.

2.4 Quantum principal component analysis

Quantum Principal Component Analysis (QPCA) [16] is an advanced quantum algorithm inspired by its classical counterpart, the Principal Component Analysis (PCA) [17], which is used for dimensionality reduction. While classical PCA is an efficient tool for reducing the complexity of large datasets, by extracting the most important features, it faces significant challenges when dealing with extremely high-dimensional data or quantum data structures. QPCA offers a quantum mechanics framework to address these challenges by exploiting the principles of quantum computation, enabling faster extraction of essential features, espe-

cially from quantum systems. This section covers hybrid quantum-classical models, and its integration with machine learning architectures.

The first step in Quantum Principal Component Analysis is the preparation of the quantum state from the given input data. Classical data, such as vectors or matrices, needs to be encoded into quantum states to leverage the benefits of quantum computations. This is usually done using quantum feature maps, whichencodeclassical data into quantum states.

2.4.1 Process

Classical input data $\mathbf{x}_i = (x_1, x_2, \ldots, x_n)$, where i denotes the index of the i-th data sample in the dataset, is converted into a quantum state $\psi(x_i)$ [18]. This transformation is achieved using quantum circuits that apply unitary operations to qubits. For example, using rotation gates like R_X and R_Z, we map each classical data point into an amplitude of the quantum state. This allows us to represent classical data in a higher-dimensional space known as a *Hilbert space*, a complete vector space equipped with an inner product that supports quantum state evolution and superposition. .

Quantum superposition allows us to encode multiple classical data points simultaneously in a quantum system. This is advantageousbecause it enables parallel processing of data, making QPCA potentially faster than classical PCA when dealing with large datasets.

Once the data is transformed into quantum states, it is represented using a density matrix ρ, which captures the ensemble of quantum states representing the data. The density matrix plays a critical role in subsequent steps, as it encapsulates the statistical mixture of different quantum states and serves as the input for the covariance matrix calculation.

The density matrix ρ is defined as :

$$\rho = \sum_i p_i \psi(x_i)\psi(x_i), \tag{2.1}$$

where p_i represents the probability associated with each quantum state, and $\psi(x_i)$ corresponds to the quantum state generated from the classical input data x_i. The term $\psi(x_i)\psi(x_i)$ is the outer product of the quantum state with itself, which defines a pure state density operator. This formulationcaptures the probabilisticnature of thequantum system by weighting each quantum state with its respective probability.

The second phase of QPCA involves calculating the quantum covariance matrix, analogous to the classical covariance matrix in traditional PCA. In classical PCA, the covariance matrix captures the variance and correlation between different features of the data. Similarly,

in QPCA, the quantum covariance matrix measures the variance and correlations between different quantum states.

Quantum Covariance Matrix Definition is the covariance matrix in QPCA, constructedusing the quantum density matrix ρ and expressed as :

$$C_{ij} = \langle \psi(x_i) | \rho | \psi(x_j) \rangle. \tag{2.2}$$

This covariance matrix reflectsthe relationships between different quantum states, similar to how the classical covariance matrix captures correlations between features in classical data.

This covariance matrix reflectsthe relationships between different quantum states, similar to how the classical covariance matrix captures correlations between features in classical data.

Since the elements of the quantum covariance matrix are quantum mechanical observables, they cannot be directly accessed because quantum measurements probabilistically collapse the quantum state. Instead, quantum measurement techniques are used to estimate the expectation values corresponding to the matrix elements. This requires multiple runs of the quantum circuit to gather measurement statistics, which are then averaged to reconstruct the covariance matrix.

A quantum circuit is a sequence of quantum gates applied to qubits, designed to manipulate quantum information. Each gate transforms the state of the qubits according to the rules of unitary evolution, and measurement at the end collapses the state to obtain classical outcomes.

2.5 Variational quantum eigensolver (VQE) in quantum machine Learning

In quantum machine learning, one of the most promising techniques for addressing quantum many-body problems is the Variational Quantum Eigensolver (VQE) [19]. VQE is a hybrid quantum-classical algorithm that combines the computational strengths of quantum mechanics with classical optimization techniques to solve for the lowest energy of a Hamiltonian system. .

The Variational Quantum Eigensolver (VQE) was initially proposed to address one of the core challenges in quantum mechanics that was finding the ground state energy of a quantum system. Ground state energy is crucial for understanding the physical properties of molecules and materials. The reason is the ground state energy shows amount of energy is using by quantum circuitand if ground state energy is high it means the quantum circuit is not optimized. This process is modeled by solving the eigenvalue problem of the system's Hamiltonian.

The goal of VQE is to find the lowest energy eigenvalue, also known as the ground state energy, which corresponds to the most stable configuration of the system.

The primary challenge is that classicalcomputers struggle to handle the complexity of quantum systems due to the exponential growth in computational resources required to simulate large numbers of qubits. VQE circumvents this issue by using quantum computers to handle the quantum part of the computation, while a classical optimizer is employed to minimize the energy.

Working principle of VQE

VQE employs a hybrid quantum-classical approach [2]. The quantum computer is responsible for preparing a quantum state and measuring its energy, while the classical optimizer updates the parameters of the quantum circuit to minimize the energy. This is done iteratively until the energy converges to a minimum.

The procedure of VQE can be broken down into the following key steps :

- **Quantum state preparation :** The first step involves preparing a quantum state, $\psi(\theta_{optimal})$, that depends on a set of adjustable parameters θ. This is done using a parameterized quantum circuit, often called the Ansatz. The circuit consists of *quantum gates* (e.g., R_X, R_Z, CX), which are basic unitary operations that manipulate the state of qubits by rotating or entangling them based on the chosen parameters.
- **Measurement :** Once the quantum state has beenprepared, the expectation valueof the Hamiltonian H with respect to the state $\psi(\theta_{optimal})$ is measured. The expectation value gives an estimate of the energy $E(\theta)$.
- **Classical optimization :** After measuring the energy, the parameters θ are updated using a classical optimization algorithm, such as gradient descent or COBYLA. The goal is to find the optimal set of parameters that minimizes the energy.
- **Iterative Process** : The process of quantum state preparation, measurement, and parameter optimization is repeated until the energy converges to a minimum, indicating that the quantum circuit has found the ground state of the Hamiltonian.
- **Output :** The final output of VQE is the optimized quantum state $\psi(\theta_{optimal})$, which approximates the ground state, and the corresponding minimum energy.

Application of VQE in quantum machine learning

In Quantum-Enhanced Adaptive Neural Network with quantum gradient descent (QEANN-QGD) models, the Variational Quantum Eigensolver (VQE) plays a critical role in quantum

feature extraction. The quantum circuit used in VQE, known as the parameterized ansatz circuit, prepares quantum states whose ground state energy is measured. This energy value can be used as a discriminative feature for classical machine learning models. This process allows the hybrid quantum-classical network to capture complex quantum phenomena, such as entanglement and superposition, which are beyond the capabilities of classical algorithms.

By leveraging VQE, the model extracts quantum features that encode the properties of the data in a high-dimensional quantum state. These features can then be integrated into the classical neural network through Hybrid Feature Integration (HFI), where they are combined with classical features to enhance predictive performance. The Challenges of Quantum Covariance Matrix are that operates in exponentially large Hilbert spaces due to the superposition of quantum states. This is why quantum computers are required to efficiently compute these correlations, as classical computers would struggle with the sheer dimensionality of the quantum states.

Dirac Notation (Bra−Ket Notation)

Dirac notation is a standard formalism in quantum mechanics for representing quantum states and their inner products [20]. It uses two main components : the ket $|\psi\rangle$ and the bra $\langle\phi|$.

- A ket $|\psi\rangle$ represents a column vector (quantum state) in a Hilbert space :

$$|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}, \tag{2.3}$$

  where α and β are complex amplitudes, and the state is normalized such that $|\alpha|^2 + |\beta|^2 = 1$.
- A bra $\langle\phi|$ is the conjugate transpose (row vector) of a ket :

$$\langle\phi| = (\gamma^*, \delta^*), \tag{2.4}$$

  where $*$ denotes complex conjugation.
- The inner product between two states $\langle\phi|\psi\rangle$ gives a complex number :

$$\langle\phi|\psi\rangle = \gamma^*\alpha + \delta^*\beta, \tag{2.5}$$

  which corresponds to the probability amplitude.

— The outer product $|\psi\rangle\langle\phi|$ forms a matrix :

$$|\psi\rangle\langle\phi| = \begin{bmatrix} \alpha\gamma^* & \alpha\delta^* \\ \beta\gamma^* & \beta\delta^* \end{bmatrix}, \tag{2.6}$$

often used to construct projection operators.

Dirac notation simplifies the expression of quantum states, measurements, and linear operations in quantum computing and quantum mechanics.

No-Cloning Theorem

The No-Cloning Theorem is a fundamental result in quantum information theory that states it is impossible to create an identical copy of an arbitrary unknown quantum state [20].

There exists no quantum operation (unitary or otherwise) that can clone an arbitrary quantum state $|\psi\rangle$.

Suppose we have a quantum system in an unknown state $|\psi\rangle$ and an auxiliary blank state $|\mathbf{0}\rangle$, and we wish to construct a unitary operation U such that :

$$U(|\psi\rangle \otimes |\mathbf{0}\rangle) = |\psi\rangle \otimes |\psi\rangle. \tag{2.7}$$

for all possible states $|\psi\rangle$.

Let $|\psi\rangle$ and $|\phi\rangle$ be two arbitrary quantum states. If perfect cloning were possible, we would also have :

$$U(|\phi\rangle \otimes |\mathbf{0}\rangle) = |\phi\rangle \otimes |\phi\rangle. \tag{2.8}$$

Taking the inner product of the two results gives :

$$\langle\psi|\phi\rangle = \langle\psi|\phi\rangle^2. \tag{2.9}$$

This implies that either $\langle\psi|\phi\rangle = \mathbf{0}$ (orthogonal states) or $\langle\psi|\phi\rangle = \mathbf{1}$ (identical states), which contradicts the requirement for cloning arbitrary (non-orthogonal) quantum states.

Hilbert Space

In quantum mechanics, a Hilbert space is a complete vector space equipped with an inner product, which provides the mathematical framework for representing quantum states [20].

- A Hilbert space H is a complex vector space where each quantum state $|\psi\rangle \in H$ is represented as a vector with complex-valued components.
- The inner product between two states $|\psi\rangle$ and $|\phi\rangle$ is denoted as $\langle\phi|\psi\rangle$, and satisfies the following properties :

$$\langle\phi|\psi\rangle = \overline{\langle\psi|\phi\rangle}, \quad \langle\psi|\psi\rangle \geq 0, \tag{2.10}$$

  where equality holds if and only if $|\psi\rangle = 0$.
- Quantum states are normalized such that:

$$\langle\psi|\psi\rangle = 1. \tag{2.11}$$

  This ensures the total probability of finding the system in some state is 1.
- The Hilbert space of a single qubit is two-dimensional, denoted H_2, with computational basis states :

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{2.12}$$

- For multi-qubit systems, the overall Hilbert space is formed by the tensor product of individual qubit spaces :

$$H_{2^n} = H_2^{\otimes n}, \tag{2.13}$$

  where n is the number of qubits.

Hilbert space provides the foundation for all quantum computations and describes how quantum systems evolve, interact, and are measured.

2.6 Quantum gates

The Hadamard gate, denoted as H, is a fundamental single-qubit quantum gate that creates superposition. It maps the computational basis states $|0\rangle$ and $|1\rangle$ into equal superpositions :

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \quad H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \tag{2.14}$$

In matrix form, the Hadamard gate is represented as :

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{2.15}$$

Applying the Hadamard gate to a qubit prepares it in a superposition of basis states, which is essential for many quantum algorithms. It is often used at the beginning of circuits to initialize quantum parallelism [20].

- **RZ gates :** These gates perform rotations around the Z-axis of the Bloch sphere, introducing a phase shift to the quantum state [20]. The action of an RZ gate on a single qubit is represented by the unitary matrix

$$R_z(\theta) = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}. \tag{2.16}$$

  Applying $R_z(\theta)$ to a general qubit state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ results in

$$R_z(\theta)|\psi\rangle = \alpha e^{-i\theta/2}|0\rangle + \beta e^{i\theta/2}|1\rangle. \tag{2.17}$$

  This phase shift is crucial for encoding phase-based information into the quantum system.

- **RX gates :** These gates perform rotations around the X-axis of the Bloch sphere, modifying the amplitudes of the qubit states. The unitary matrix representation of an RX gate is

$$R_x(\theta) = \begin{bmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{bmatrix} \tag{2.18}$$

  Applying $R_x(\theta)$ to a general qubit state results in

$$R_x(\theta)|\psi\rangle = \cos(\theta/2)\alpha|0\rangle - i\sin(\theta/2)\beta|1\rangle + \cos(\theta/2)\beta|1\rangle - i\sin(\theta/2)\alpha|0\rangle. \tag{2.19}$$

  This transformation allows the quantum circuit to adjust the probability amplitudes of the qubit states, making it an essential component for learning amplitude-based features.

CNOT gates (entanglement)

The CNOT gate acts on a pair of qubits, consisting of a control qubit and a target qubit [20]. The gate flips the state of the target qubit if and only if the control qubit is in the state $|1\rangle$. Mathematically, the CNOT operation is represented by the 4×4 unitary matrix

$$\mathrm{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{2.20}$$

Applying the CNOT gate to a two-qubit system in a general state,

$$|\psi\rangle = \alpha|\mathbf{00}\rangle + \beta|\mathbf{01}\rangle + \gamma|\mathbf{10}\rangle + \delta|\mathbf{11}\rangle, \tag{2.21}$$

produces the transformation

$$\mathrm{CNOT}\,|\psi\rangle = \alpha|\mathbf{00}\rangle + \beta|\mathbf{01}\rangle + \gamma|\mathbf{11}\rangle + \delta|\mathbf{10}\rangle. \tag{2.22}$$

This transformation swaps the last two terms, flipping the target qubit when the control qubit is in state $|\mathbf{1}\rangle$.

Bloch Sphere

The Bloch sphere is a geometric model used to represent the state of a single qubit as a point on the surface of a unit sphere in three-dimensional space. In this representation, every pure qubit state corresponds to a unique position on the sphere, determined by two angles that act like spherical coordinates. The north and south poles of the sphere represent the classical basis states, while all other points correspond to quantum superpositions of these states. This visualization is particularly useful for understanding and illustrating quantum operations such as rotations, phase shifts, and general qubit transformations [20].

Single-Qubit and Multi-Qubit Systems

A qubit is the basic unit of quantum information, comparable to a classical bit but governed by the principles of quantum mechanics. Unlike classical bits that can only exist in one of two definite states, a qubit can exist in a superposition of its basis states. This property allows quantum systems to process and encode information in fundamentally different ways compared to classical systems [20].

Multi-qubit systems are formed by combining individual qubits into larger composite systems. For example, a system composed of two qubits spans a four-dimensional space, while a system of three qubits spans an eight-dimensional space, and so on. These systems exhibit complex

behaviors such as entanglement, where the state of each qubit is intrinsically linked to the others. Entanglement is a uniquely quantum phenomenon that enables powerful protocols such as quantum teleportation, superdense coding [20].

As the number of qubits increases, the size of the state space grows exponentially. An n-qubit system can represent information in a space of dimension $\mathbf{2}^n$, which provides a significant computational advantage for certain classes of problems when compared to classicalsystems[7].

2.7 Hybrid Quantum Autoencoders

Hybrid Quantum Autoencoders (HQA) are a type of neural network that combines classical and quantum computing elements to efficiently compress and reconstruct quantum data like images. By leveraging quantum circuits for encoding, HQAs can handle image patterns while maintaining lower resource requirements, when compared to fully quantum approaches. This hybrid structure enables HQAs to exploit both quantum and classical advantages, making them particularly valuable in quantum data processing and noise reduction applications. M. Srikumar et al [21] did the first study on the HQA involving technology created from equivalently classical ANNs and quantum neural networks (QNN), where data compression is optimized using PQC for better classification. The use of an encoder and decoder, as illustrated in figure 2.1, is appropriate here. The encoder takes a quantum state from a higher-dimensional Hilbert space into a lower-dimensional real vector space. At the same time, the decoder reverses the transformation to reconstruct the original quantum state.

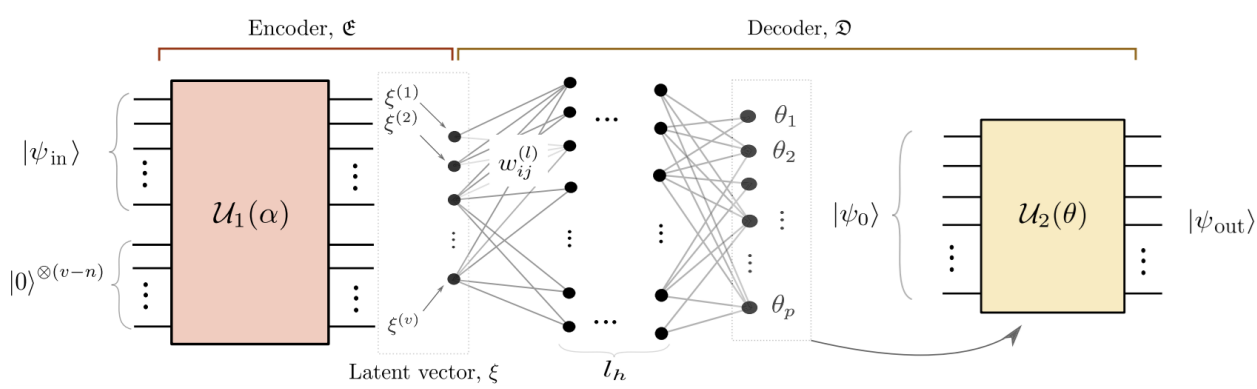


Figure 2.1 Hybrid quantum auto-encoder [21].

The encoder is implemented in this study as a parameterized quantum circuit, where the rotation gates are selected from the Pauli gates X, Y , and Z, with the rotation angles treated as trainable parameters. These gates are fundamental single-qubit operations used

to manipulate the qubit states on the Bloch sphere.

The input state to the circuit consists of an entangled quantum state ψ_{in} combined with ancillary qubits initialized in the ground state $\mathbf{0}^{\otimes(v-n)}$, where v is the total number of qubits and n corresponds to the number of data-carrying qubits. The latent vector $\boldsymbol{\xi}$ is a compressed classical representation of the quantum state, formed by measuring the expectation values of the Z operator on each qubit after the encoding. These expectation values are calculated by repeatedly measuring the same quantum circuit and computing statistical averages of the outcomes, which reflect the likelihood of measuring each qubit in state $\mathbf{0}$ or $\mathbf{1}$.

This latent representation enables efficient clustering or classification in a lower-dimensional classicalspace. Hence, the classification of quantum data can be performed without the need for full quantum state tomography.

The Hybrid Quantum Autoencoder (HQA) is particularly effective in semi-supervised learning tasks. When trained on a set of Gaussian amplitude-encoded quantum states, the model achieves an accuracy of 84% in clustering tasks and 93% in classification tasks. This demonstrates the feasibility of capturing essential quantum features within a classical latent space and highlights the effectiveness of HQA for quantum data analysis.

This hybrid framework allows compression of quantum states while simultaneously enabling the use of classical machine learning algorithms for downstream tasks, thus overcoming limitations of traditional quantum autoencoders and offering new potential for quantum machine learning.

2.8 Quantum multiclass classifier (QMCC)

A Quantum Multiclass Classifier (QMCC) is a quantum algorithm designed to categorize data into multiple classes, extending the binary classification capabilities of quantum systems. By leveraging quantum states and measurement outcomes, these classifiers can encode and distinguish complex data patterns like images efficiently, offering potential speed and accuracy advantages over classical counterparts. Quantum multiclass classifiers are particularly useful in fields like image recognition and natural language processing, where data often belongs to more thantwocategories. This method developed by AvinashChalumuri et al [22], the Quantum MultiClass Classifier employs a variational circuit as shown in Figure 2.2 to classify datasets with multiple classes. The model uses a hybrid approach by combining classical optimization techniques with quantum processing, specifically to classify data with four features.

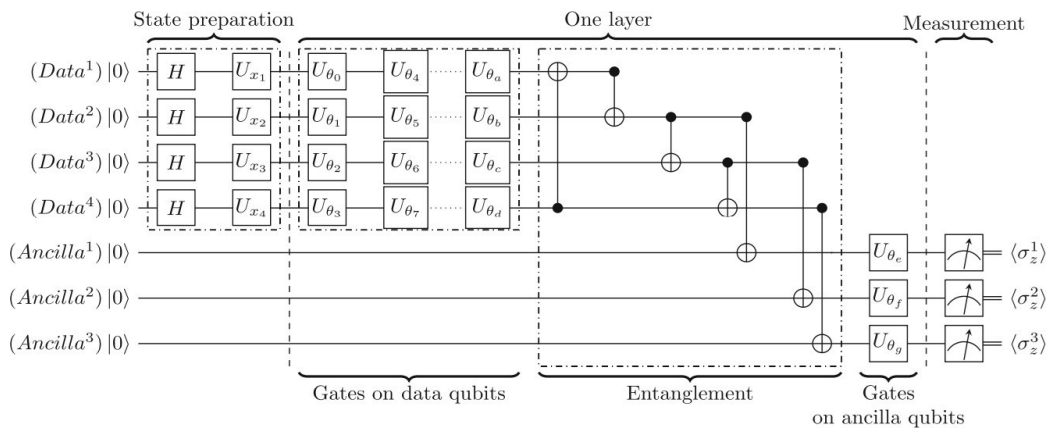


Figure 2.2 Quantum variational circuit model [22].

Starting from a configuration with all qubits set in the $|\mathbf{0}\rangle$ state, the variational circuit applies Hadamard gates to each qubit to create an equal superposition. Following this step, the classical data is encoded into the quantum state through unitary operations wherein each qubit will experience some transformation dictated by a unitary square matrix. The encoded information is thereby rendered as a composite quantum state. Starting from an initial configuration where all qubits are set to the $\mathbf{0}$ state, the variational circuit first applies Hadamard gates to each qubit.

Following this, classical data is encoded into the quantum state using unitary operations, typically parameterized rotation gates. These operations transform each qubit's state according to learnable parameters, rendering the input as a composite quantum state.

The variational quantum circuit, also called a parameterized quantum circuit (PQC), is structured in multiple layers, each consisting of trainable rotation gates on data and ancillary qubits. These parameters are optimized using a classical optimizer in a hybrid quantum-classical loop.

To introduce entanglement, Controlled-NOT (CNOT) gates are applied between qubits. CNOT gates are two-qubit operations that flip the target qubit's state conditional on the control qubit's value. This step enables the circuit to capture correlations between features.

At the final stage, selected qubits—typically the ancillary ones—are measured, and their outcomes are used in downstream classical processing (e.g., for classification or regression tasks). Starting from an initial configuration where all qubits are set to the $\mathbf{0}$ state, the variational circuit first applies Hadamard gates to each qubit. A Hadamard gate is a single qubit operation that creates a uniform superposition between the $\mathbf{0}$ and $\mathbf{1}$ states, enabling

quantum parallelism.

Next, classical data is encoded into the quantum state using parameterized unitary operations, typically rotation gates, that transform each qubit based on learnable parameters. The resulting quantum state encodes the information in a distributed and entangled format.

After processing through several layers, the ancillary qubits are measured. The measurement results are then fed into a classical softmax function [23], which converts the measurement outcomes into a probability distribution over the possible class labels. This final step enables the use of the quantum circuit output for classification tasks in hybrid quantum-classical models.

2.9 Support Vector Machines With a Quantum Kernel Estimator (QSVM-Kernel)

Support Vector Machines with a Quantum Kernel Estimator (QSVM-kernel) use quantum computing to estimate kernel functions, enhancing the SVM's ability to separate complex data patterns like images. The quantum kernel exploits the high-dimensional feature space created by quantum states, allowing for more effective classification of non-linear data. QSVM-kernel approaches are especially promising for tasks involving large and intricate datasets, as they can potentially outperform classical SVMs in both speed and accuracy. S. L. Wu et al [24] proposed method whichdescribes a quantum-enhanced approach to support vector machines. The algorithm leverages quantum state space as the feature space for efficiently computing kernel inputs. In non-linear fashion, this method maps classical data into a quantum state using a quantum circuit, that prepares a high-dimensional feature space that is difficult to attain with classical methods.

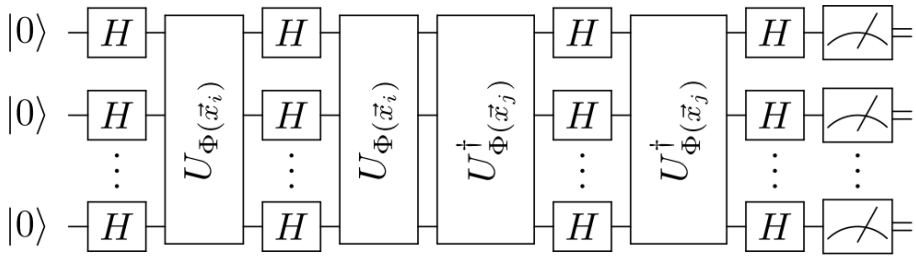


Figure 2.3 Quantum circuit of support vector machine [24].

The quantum circuit, as illustrated in Figure 2.3, consists of two repeated layers. Each layer begins with Hadamard gates applied to all qubits to create equal superposition states, fol-

lowed by a unitary transformation $U_{\Phi(\vec{x})}$ that encodes the classical input data into quantum states. This processmaps the data into a high-dimensional Hilbert spacewith dimensionality 2^N, where N is the number of qubits.

During training, the kernel values are computed on a quantum computer and used to construct a decision boundary typically a hyperplane that separates the data classes in this transformed space. This study applies the Quantum Support Vector Machine with Quantum Kernel Estimator (QSVM-Kernel) algorithm to classify high-energy physics events from the Large Hadron Collider, particularly in the top–antitop quark Higgs boson production channel ($t\bar{t}H$). The algorithm maps classical input data $\vec{x}$ into quantum states using a data-encoding quantum feature map $U_\Phi(\vec{x})$ on N qubits, resulting in the quantum state $|\Phi(\vec{x})\rangle = U_\Phi(\vec{x})|0\rangle^{\otimes N}$. The similarity (kernel) between two data points is computed as the squared inner product of their respective quantum states.

To evaluate this kernel $k(\vec{x}_i, \vec{x}_j) = |\langle\Phi(\vec{x}_i)|\Phi(\vec{x}_j)\rangle|^2$, the authors construct a quantum circuit involving Hadamard gates and parameterized single-qubit rotations along with Controlled-NOT (CNOT) entangling operations. This circuit is executed on quantum simulators and IBM's superconducting quantum hardware. The feature map is specifically designed to be hard to simulate classically while remaining feasible on Noisy Intermediate-Scale Quantum (NISQ) devices.

During training, the quantum computer is only used to compute kernel entries, while the Support Vector Machine (SVM) optimization (i.e., finding the separating hyperplane) is performed on a classicalcomputer. Input features are reducedfrom 23 kinematic variables to N variables using Principal Component Analysis (PCA), aligning the feature space dimension with the number of available qubits. This hybrid quantum-classical architecture achieves classification performance comparable to that of classical Support Vector Machines (SVMs) and Boosted Decision Trees, demonstrating the practical viability of quantum kernel methods for high-energy physics event classification, even when operating under realistic hardware noise conditions.

The use of quantum kernels in this approach has shown promising results in improving classification performance, particularly when using projected quantum kernel methods. These methods offer enhanced capacity for capturing complex correlations in data, often achieving lower prediction error compared to certain classical models. Overall, the results highlight the potential of quantum kernel methods and variational circuits in addressing challenging classification tasks that are difficult for classical techniques alone.

2.10 Quantum fully self-supervised neural network (QFS-Net)

The Quantum Fully Self-Supervised Neural Network (QFS-Net) [25] introduces a novel quantum neural network model that leverages qubits, which are multi-level quantum systems (with more than two states). In this model, each qubit is represented by multiple basis states. For instance, a qubit is a three-level quantum system used as the fundamental processing unit in the QFS-Net. The QFS-Net applies a fully self-supervised learning approach, where the information is processed using a network of qubit neurons connected by Hadamard gates. The qubit neurons update their states using a transformation gate and rotation angles based on the difference in information between neighboring qubit neurons. This mechanism guides the self-propagation and counter-propagation of quantum states across the network layers without external supervision.

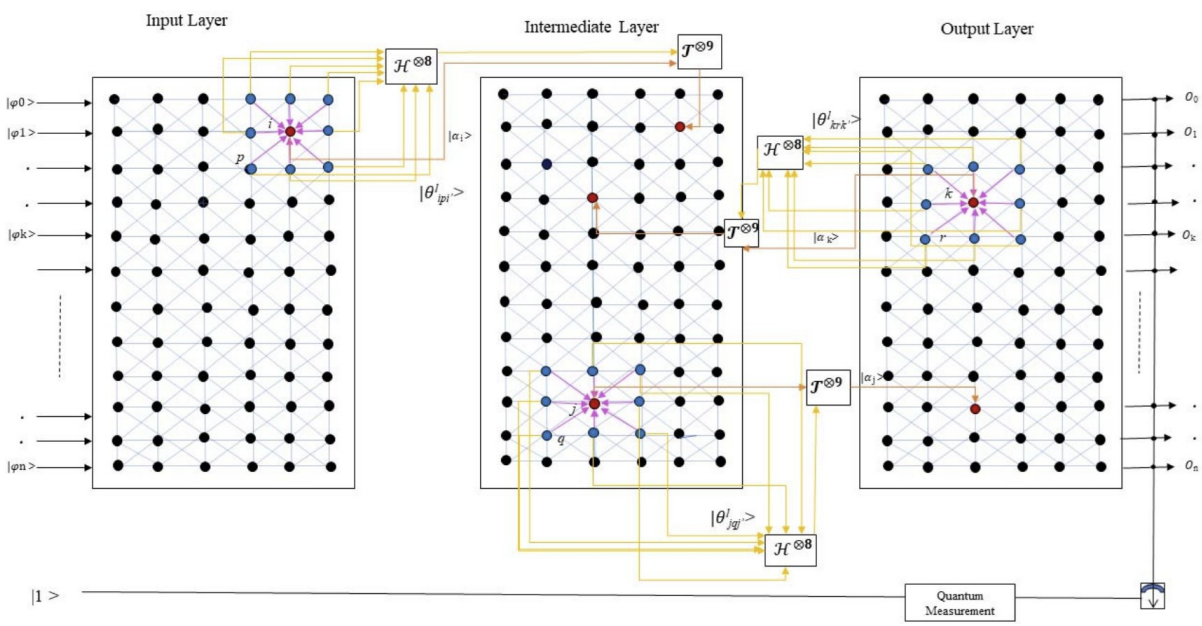


Figure 2.4 QFS-Net [25].

The QFS-Net architecture, as illustrated in Figure 2.4, was validated on brain tumor segmentation tasks using MRI images from the Cancer Imaging Archive (TCIA). QFS-Net consists of three main layers, an input layer, an intermediate (or transformation) layer, and an output layer, each constructed from grid-based quantum processing units. The circuit includes operators such as $H^{\otimes 8}$, representing an 8-qubit Hadamard transformation to create superpositions, and $T^{\otimes 9}$, denoting the application of the T gate across 9 qubits to introduce non-linearity and phase shifts.

Compared to classical deep learning models such as U-Net and URes-Net, QFS-Net achieved competitive performance in terms of similarity and segmentation accuracy. These results highlight the potential of quantum fully self-supervised networks in automating complex medical imaging tasks with minimal human intervention.

2.11 Quantum Deep Convolutional Neural Networks (QDCNN)

Quantum Convolutional Neural Networks (QCNNs) are a type of neural network that integrate quantum computing into the convolutional layers aimed at capturing complex spatial patterns in data like image. By leveraging quantum operations, QCNNs can process high-dimensional inputs with potentially fewer resources than classical deep convolutional networks. This quantum adaptation is especially promising in fields requiring intricate pattern recognition, as it can offer significant computational savings while maintaining accuracy, opening new possibilities in quantum-enhanced image and signal processing. The most important paper [26] explores a quantum deep convolutional neural network (QDCNN) model for image recognition, built using a quantum parameterized circuit. Similar to the classicaldeepconvolutional neural network (DCNN), the QDCNN consists in a series of quantum convolutional layers followed by a quantum classification layer. The proposed QDCNN model begins by encoding classical image data into quantum states. Quantum convolutional layers then apply parameterized unitary operations and inner product computations via quantum arithmetic. Feature mappings are transformed using quantum phase estimation and nonlinear activation is realized through Taylor-expanded circuits. Finally, a quantum-classical hybrid algorithm updates parameters via classical optimization, enabling image classification through quantum measurement.

The training of the QDCNN is inspired by variational quantum algorithms, using a quantum–classical hybrid approach to update the parameters effectively. An analysis of the network's complexity shows that the proposed QDCNN model, , offers exponential speedup compared to its classical counterpart. Numerical simulations using the MNIST and the GTSRB datasets demonstrate the feasibility and effectiveness of the model, validating its potential for practical applications.

2.12 Long short-term memory neural networks

In classical deep learning, Long Short-Term Memory (LSTM) networks are designed to capture both short-term and long-term patterns in sequential data. An LSTM cell typically includes three main inputs : the current data point at a specific time step, the hidden state

from the previous step, and the cell state from the previous step. These inputs make the LSTM process information over time, making it suitable for tasks such as time-series forecasting and natural language processing.

The LSTM cell uses three control gates—input, forget, and output gates—that manage the flow of information. Each gate is equipped with activation functions, such as the sigmoid or hyperbolic tangent, to determine how much information should be allowed into the cell, retained, or outputted. This gating mechanism enables the LSTM to maintain relevant information over extended sequences, making it practical for tasks requiring memory of past data points [27].

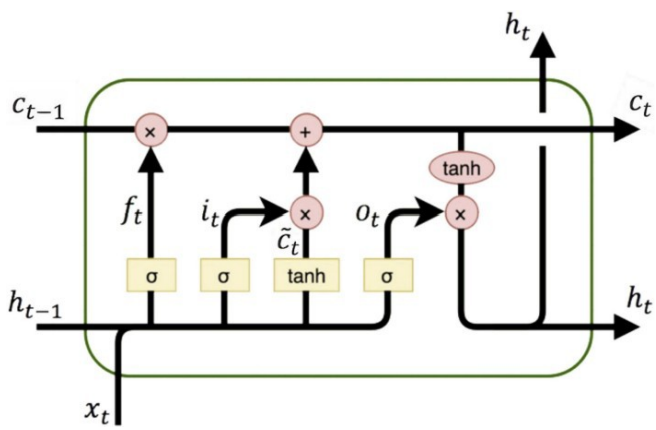


Figure 2.5 LSTM cell [27].

A hybrid architecture is proposed in, Figure 2.5. It includes two layers with 32 LSTM cells each, followed by two fully connected layers, where the second layers contains 10 neurons and connects to the QNN. The QNN is constructed using the IQP(Instantaneous Quantum Polynomial-time) anstaz and strongly entangling layers, and it includes a final output layer that operates classically. The term IQP refers to a restricted model of quantum computation where all the quantum gates in the circuit are diagonal in the computational basis and are applied simultaneously that is, without sequential dependence. IQP circuits typically consist of a layer of Hadamard gates, followed by a series of commuting gates (e.g., Z, CZ, or T gates), and then another layer of Hadamard gates. The proposed Quantum LSTM (QLSTM) algorithmreplicatesthe structure of aclassicalLSTMcellusingreversiblequantum gates. Inputs $\mathbf{x}_t$, $\mathbf{h}_{t-1}$, and $\mathbf{c}_{t-1}$ are encoded via basis encoding in n-qubit registers using Q notation, allowing integer and fractional representation. Core LSTM operations are addition, multiplication, and activation functions implemented as quantum subroutines : quantum addition via conditional bitwise incrementation, quantum multiplication via controlled-Toffoli

networks, and nonlinear activations (sigmoid, tanh) through Boolean circuit design using product-of-sums logic. Ancilla qubits and uncomputation techniques ensure reversibility and reuse of quantum memory. The algorithm was tested on IBM's simulator, confirming that the output states $\mathbf{h}_t$ and $\mathbf{c}_t$ match their classical counterparts. This method lays the groundwork for future quantum RNN architectures on fault-tolerant hardware.

Although IQP circuits are not universal for quantum computation, they are believed to be hard to simulate classically, making them useful in demonstrating quantum advantage in specific tasks such as quantum sampling or certain machine learning applications.

2.13 Recurrent quantum neural networks (RQNN)

Recurrent Quantum Neural Networks (RQNNs) extend the principles of recurrent neural networks into the quantum domain, allowing information to cycle through the network layers for capturing temporal patterns in sequential data. By leveraging quantum superposition and entanglement, RQNNs offer potential advantages in processing complex time-series data efficiently, making them promising for applications like speech recognition, language processing, and dynamic system modeling. The study by L. Gyongyosi as illustrated in Figure 2.6 [28], that focuses on optimizing the training process for gate-model quantum neural networks (QNNs), introducing a constraint-based optimization method aimed at both non-recurrent and recurrent network structures.This work introduces a formal training optimization framework for gate-model quantum neural networks (QNNs), referred to as QNNQG and its recurrent extension RQNNQG. In these architectures, the quantum computation proceeds through a chain of parameterized unitary gates $U_i(\theta_i)$ acting on an input quantum state, with a readout qubit used for measurement. Classical side information (such as gradients and error terms) is used to optimize gate parameters through a hybrid quantum-classical approach.

The authors define an objective function $\mathbf{f}(\vec{\theta})$ that captures the distance between predicted and true labels over a training set. The QNN learns by minimizing this function via gradient descent, where gradients are computed using measured Pauli observables from quantum hardware or simulators. For recurrent quantum networks (RQNNQG), past measurement outcomes are fedback to influence future computations, enabling time-aware optimization.

To formalize learning in these networks, the authors adopt a constraint-machine model. They model the environment of the QNN as a directed acyclic graph (DAG), where nodes represent unitary gates and edges represent information flow. Learning is posed as solving Euler–Lagrange equations under structural and environmental constraints. The gradients are calculated using chain rule-like dependencies between unitary operations, and the optimiza-

tion complexity scales with the number of gate parameters.

In the recurrent version (RQNNQG), side information is allowed to propagate backward in time. This structure mimics classical recurrent neural networks (RNNs) and supports backpropagation through time. As proven in the paper [28], such recurrent backpropagation is optimal for gradient-based training in these networks. The proposed approach efficiently optimizes parameters by incorporating classical side information and environmental constraints. Using simulations, the paper shows that this method effectively minimizes the loss function and updates gate parameters, making it a practical training solution for near-term quantum hardware applications.

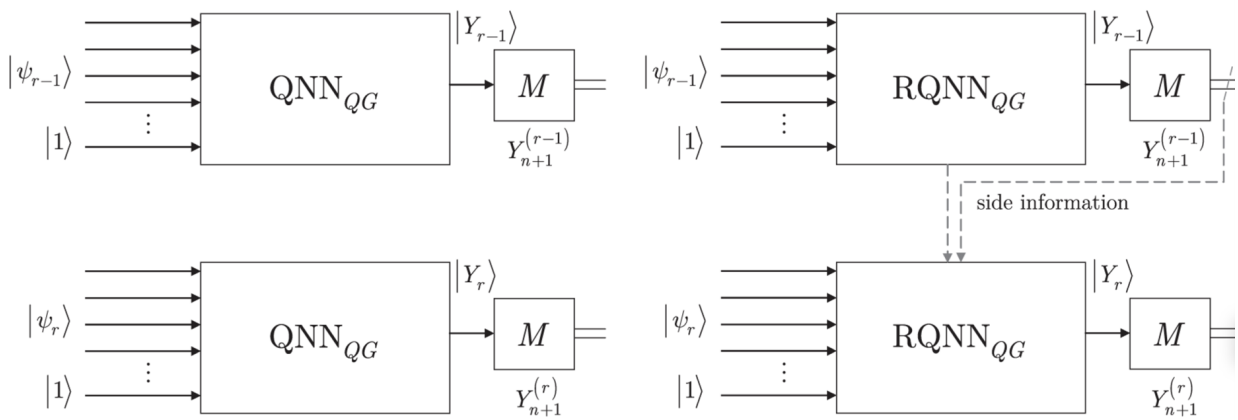


Figure 2.6 Schematic representation of QNNs and RQNNs [28].

2.14 Quantum Recurrent Encoding–Decoding Neural Networks (QREDNN)

The Quantum Recurrent Encoder–Decoder Neural Network (QREDNN) is designed to enhance sequential data processing tasks, such as image segmentation, by capturing temporal dependencies using quantum operations. This architecture forms the basis for optimizing quantum gate-based neural networks in both recurrent and non-recurrent configurations.

As illustrated in Figure 2.7, the model incorporates trainable quantum recurrent units within an encoder–decoder structure. QGRU (Quantum Gated Recurrent Units) are quantum analogs of classical gated recurrent units that encode temporal features using parameterized quantum gates and entanglement. This enables the network to learn complex time-dependent patterns in quantum data.

To address optimization in these networks, a constraint-based method is proposed [29], which adjusts quantum gate parameters using classical side information and environmen-

tal constraints.

Simulations using the structure shown in Figure 2.7 demonstrate that this approach effectively minimizes the loss function and updates the gate parameters during training, making it a suitable framework for near-term quantum computing platforms.

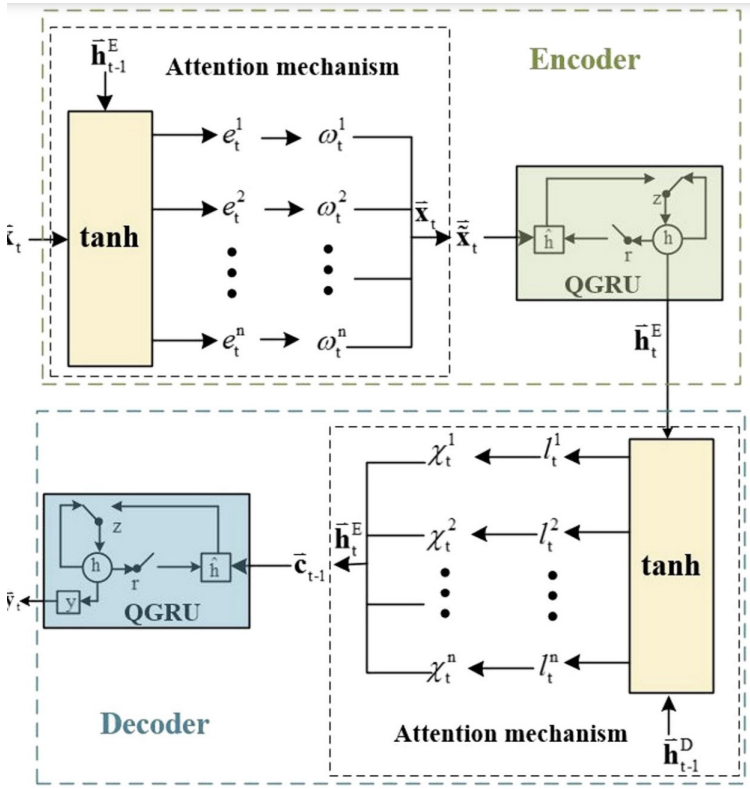


Figure 2.7 The structure of quantum recurrent encoder–decoder neural network [29].

The Quantum Recurrent Encoder–Decoder Neural Network (QREDNN) algorithm proposes a hybrid quantum-classical neural architecture that enhances the traditional encoder–decoder framework by integrating Quantum Gated Recurrent Units and dual attention mechanisms [29]. Each QGRU replaces classical neurons with quantum neurons, where inputs and weights are represented by quantum states using rotation matrices on the Bloch sphere. Specifically, quantum rotation gates around the x, y, and z axes are used to model activation and update functions, exploiting the periodic and high-dimensional nature of quantum operations for finer solution space traversal.

In the encoder, input sequences are mapped to quantum states and passed through QGRU, where quantum neurons apply rotations and projections to compute gate activations. An attention mechanism dynamically weights input components by computing relevance scores based on hidden states, allowing the model to focus on significant temporal patterns and

suppress noise. Similarly, the decoder uses QGRU with an independent attention mechanism to refine sequence generation.

To optimize the network, the Levenberg–Marquardt [29] algorithm is employed instead of traditional gradient descent, enabling fast and stable updates of quantum rotation angles and attention parameters. The QREDNN is trained on Denoised Fuzzy Entropy features extracted from vibration signals of rotating machinery, and its output predicts future degradation trends.

The algorithm's strength lies in its ability to combine quantum parallelism and attention-driven dynamic weighting. This results in improved nonlinear approximation capacity, higher generalization performance, and faster convergence compared to classical models like Gated Recurrent Unit Neural Network or Deep Neural Network with Recurrent Encoder–Decoder. The experimental results demonstrate that the network achieves state-of-the-art accuracy and robustness for time-series degradation trend prediction.

CHAPTER 3 QUANTUM ENHANCED ADAPTIVE NEURAL NETWORK

3.1 Quantum Gradient Descent

Optimization plays a fundamental role in various disciplines, including machine learning [30], finance [31], and biomedical research[32]. In classicalmachine learning, gradient descent is a widely used optimization technique that iteratively updates parameters to minimize a given loss function, allowing models to improve their predictive performance.

Quantum Gradient Descent (QGD) extends this concept to quantum computing by optimizing parameters within quantum circuits. The fundamental distinction between classical and quantum gradient descent lies in how gradients are computed and applied. While classical optimization relies on numerical differentiation or backpropagation, QGD exploits superposition and entanglement to process multiple gradient components simultaneously, potentially improving computational efficiency.

In hybrid quantum–classical optimization, quantum circuits are executed repeatedly, and their measurement outcomes are passed to classical optimization algorithms to update model parameters [33]. However, the interaction between quantum and classical components presents unique challenges, particularly in ensuring stable convergence and mitigating the impact of quantum noise.

Effectively addressing theseissues is essential for the successful integration of quantum gradient descent (QGD) into quantum machine learning frameworks. Figure 3.1 illustrates the gradient computation pipeline, which combines classical and quantum stages. This hybrid optimization process ensures that quantum parameters are updated efficiently to minimize the loss function.

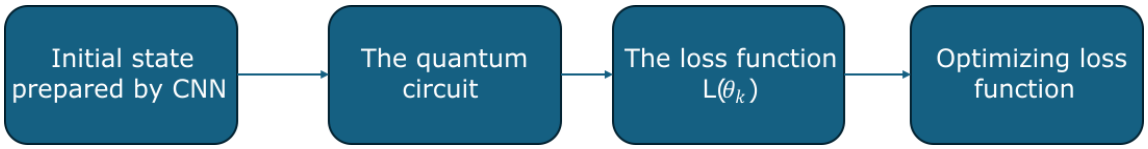


Figure 3.1 Schematic of calculating the gradients

The process consists of the following key steps :

— **Initial state preparation** : The process begins with the classical convolutional neural

network (CNN) preparing the initial feature representation. These extracted features are subsequently encoded into a quantum state, forming the input to the quantum circuit.

- **Quantum circuit processing** : The quantum circuit processes the input state using a parameterized quantum circuit (PQC), where quantum gates act on the qubits based on trainable parameters θ_k. The circuit evolution determines how the quantum-enhanced features are transformed.
- **Loss function computation** : After measurement, the quantum circuit outputs are evaluated against a predefined loss function $L(\theta_k)$. This function quantifies the error between the predicted quantum-enhanced representation and the expected outcome.
- **Optimization of the loss function** : The computed loss value is then used to update the quantum circuit parameters. A classical optimizer, such as Quantum Gradient Descent (QGD) or Adam, adjusts θ_k iteratively based on the gradient information, ensuring convergence toward an optimal solution.

This hybrid quantum-classical gradient computation approach enables the model to leverage quantum feature extraction while utilizing classical optimization techniques for efficient parameter updates. The iterative process refines the quantum circuit's parameters, improving the overall learning performance of QEANN-QGD.

3.1.1 Quantum Algorithm for Gradient Estimation

The quantum gradient descent process is illustrated in Figure 3.2.

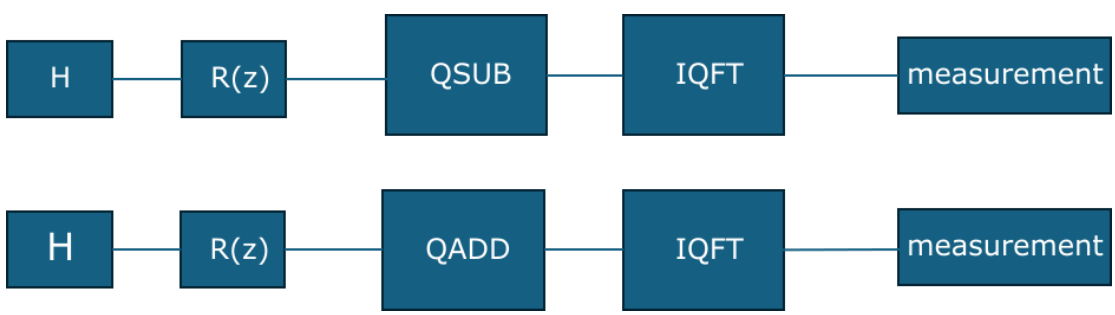


Figure 3.2 Schematic quantum gradient descent

This diagram illustrates how quantum gates, including the Hadamard gate, along with specialized operations such as Quantum Addition (QADD) and Quantum Subtraction (QSUB),

can be integrated with the Inverse Quantum Fourier Transform (IQFT) and measurement processes to facilitate complex quantum computations.

The Hadamard gate (H) [20] as fundamental in quantum computing as it generates superposition, allowing a qubit initialized in $\mathbf{0}$ or $\mathbf{1}$ to transition into an equal-probability state of both $\mathbf{0}$ and $\mathbf{1}$. This property enables quantum algorithms to explore multiple computational paths simultaneously, increasing efficiency over classical counterparts.

Quantum Addition (QADD) and Quantum Subtraction (QSUB) are arithmetic operations designed to perform controlled addition and subtraction directly on quantum states. Unlike classical arithmetic, these operations leverage quantum interference and entanglement, enabling efficient encoding of numerical computations within quantum circuits. These operations are particularly useful in quantum optimization problems and algorithms requiring iterative numerical adjustments.

The Inverse Quantum Fourier Transform (IQFT) is a key transformation in many quantum algorithms, including Shor's algorithm for factoring large numbers. It is the inverse operation of the Quantum Fourier Transform (QFT) and it is used to return quantum states from the frequency domain back to the computational basis. IQFT plays a crucial role in extracting useful information from quantum states after they have undergone quantumcomputations.

Finally, measurement is the process of collapsing a quantum state into a classical outcome. Since quantum states exist in a superposition of multiple possibilities, measurement forces the system to adopt a single classical value. This step is essential in quantum computing, as it bridges the quantum and classical worlds by providing interpretable results that can be used for further decision-making or optimization.

These operations play a fundamental role in quantum algorithms. They support key tasks such as state manipulation and numerical computation. Moreover, they enable the efficient extraction of quantum information, which is essential for bridging quantum computations with classical post-processing.

When we measure the loss function, the Taylor expansion intervene ; and for infinitesimal quantities, can be expressed as [2] :

$$\mathbf{f(x) \approx f(x_0) + \nabla f(x_0) \cdot (x - x_0)}, \tag{3.1}$$

For an objective function with d variables, we begin by creating a quantum state that places the system into a superposition over all possible variable combinations. This enables the simultaneous computation of gradients with respect to all variables by leveraging quantum parallelism, where the system processes multiple inputs through superposition in a single

operation.

We represent this superposition state as [20] :

$$|\psi\rangle = \frac{1}{\sqrt{N^d}} \sum_{\delta=0}^{N-1} |\delta\rangle . \tag{3.2}$$

In this expression, N denotes the number of discretization levels per variable, representing how many distinct values each variable can take. The parameter d indicates the total number of variables in the objective function. The quantity N^d corresponds to the total number of possible variable combinations across all d dimensions. Finally, $|\delta\rangle$ represents a computational basis state typically expressed in binary—that identifies one of the N^d possible input configurations.

For example, if $N = 2$ and $d = 3$, then $N^d = 8$, and the computational basis states range from $|000\rangle$ to $|111\rangle$, each representing a unique configuration of the input space.

The state $|\psi\rangle$ is thus an equal superposition over all possible inputs, and serves as the starting point for quantum gradient computation. The use of this state enables quantum circuits to process and differentiate across all dimensions simultaneously.

To achieve the desired quantum state for the constructed approximation, we apply a quantum adder to shift the previously prepared equal-weight superposition state by a fixed increment Δx The transformation occurs by applying a unitary phase gate with phase $e^{2\pi i \left(\frac{N}{m}\right) \nabla f(\Delta x)}$ to each state $|\delta + \Delta x\rangle$, encoding the gradient $\nabla f(\Delta x)$. This results in the translated superposition [34] :

$$\frac{1}{\sqrt{N^d}} \sum_{\delta=0}^{N-1} |\delta + \Delta x\rangle = \frac{1}{\sqrt{N^d}} \sum_{\delta=0}^{N-1} e^{2\pi i \left(\frac{N}{m}\right) \nabla f(\Delta x)} \frac{\delta}{N} |\delta + \Delta x\rangle , \tag{3.3}$$

This expression incorporates a phase factor that encodes gradient information into the quantum state. By assigning a unique phase to each component of the superposition, the circuit enables parallel estimation of the gradient across all variable combinations through quantum interference.

$$e^{\frac{2\pi i}{m} \nabla f(\Delta x) 0} |1\rangle + e^{\frac{2\pi i}{m} \nabla f(\Delta x) 1} |2\rangle + e^{\frac{2\pi i}{m} \nabla f(\Delta x) 2} |3\rangle + e^{\frac{2\pi i}{m} \nabla f(\Delta x) 3} |4\rangle. \tag{3.4}$$

Next, we build an approximation that incorporates f(x) in phase and apply it to the shifted superposition state, yielding

$$\frac{1}{\sqrt{N^d}} \sum_{\delta=0}^{N-1} |\delta\rangle \xrightarrow{O_f} \frac{1}{\sqrt{N^d}} \sum_{\delta=0}^{N-1} e^{2\pi i \left(\frac{N}{ml}\right) f(x)} |\delta\rangle . \tag{3.5}$$

In this step, we apply a quantum oracle (O_f) that encodes the output of a classical function $f(x)$ into the phase of a quantum state. This process transforms an initial uniform superposition into a phase-encoded state, where the function value influences the phase of each basis state.

In the expression involving the phase term :

- $f(x)$ is a real-valued objective (or cost) function defined over the input domain.
- N is the number of discretization levels per variable.
- m and l are scaling constants introduced to ensure the resulting phase $\frac{N}{ml} f(x)$ lies within $[0, 1)$, which is necessary for the correct interpretation of the exponential term $e^{2\pi i(\cdot)}$. These constants are typically selected such that $\frac{N}{ml} \max f(x) < 1$ to avoid phase wrapping and ensure numerical stability.

To achieve the desired phase-encoded state, we apply a quantum oracle that modifies the phase of each basis state based on the dot product between the index vector δ and the gradient ∇f. Specifically, for a multi-dimensional $\delta = (\delta_1, \delta_2, \ldots, \delta_d)$ and $\nabla f = \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_d}\right)$, the inner product $\delta \cdot \nabla f$ expands to $(\delta_1)(\nabla f)_1 + (\delta_2)(\nabla f)_2 + \cdots + (\delta_d)(\nabla f)_d$.

This leads to the expression :

$$\frac{1}{\sqrt{N^d}} \sum_{\delta=0}^{N-1} e^{\frac{2\pi i}{m}[\delta_1(\nabla f)_1 + \delta_2(\nabla f)_2 + \cdots + \delta_d(\nabla f)_d]} |\delta + \Delta x\rangle \tag{3.6}$$

To extract the phase information, it is necessary to shift the superposition state back to the original equal-weight position. For this, a separable quantum subtractor is required, which results in :

$$\frac{1}{\sqrt{N^d}} \sum_{\delta=0}^{N-1} e^{\frac{2\pi i}{m}[\delta_1(\nabla f)_1 + \delta_2(\nabla f)_2 + \cdots + \delta_d(\nabla f)_d]} |\delta\rangle . \tag{3.7}$$

Next, a quantum subtractor is applied to shift the data register back to its original form, while retaining the phase encoding in the ancilla register. For our network that has 4 qubits, we will have :

$$\frac{1}{\sqrt{N^d}} \sum_{\delta=0}^{N-1} e^{2\pi i \left(\frac{N}{ml}\right) \nabla f(\Delta x)} |\delta + \Delta x\rangle . \tag{3.8}$$

To get back to the original basis, we apply the quantum subtractor to remove the shift by

Δx ;

$$\frac{1}{\sqrt{N^d}} \sum_{\delta=0}^{N-1} e^{2\pi i \left(\frac{N}{m}\right) \nabla f(\Delta x) \frac{l}{N}} |\delta\rangle \tag{3.9}$$

By using the quantum subtractor, we effectively transform the quantum state into a form that is ready for the Quantum Fourier Transform. By applying the Inverse Quantum Fourier Transform, we obtain a quantum state that encodes the gradient information ;

$$\frac{N}{m}(\nabla f(\Delta x))_1 \cdots \frac{N}{m}(\nabla f(\Delta x))_i \cdots \frac{N}{m}(\nabla f(\Delta x))_d \quad , \tag{3.10}$$

where the (i) subscript refers to the ith component of the gradient.

We then apply the oracle to embed the gradient information into the phase of the quantum state,

$$\frac{1}{\sqrt{N^d}} \sum_{\delta=0}^{N-1} e^{2\pi i \left(\frac{1}{m}\right)[\delta_1 (\nabla f)_1 + \delta_2 (\nabla f)_2 + \cdots + \delta_d (\nabla f)_d]} |\delta - \Delta x\rangle \quad , \tag{3.11}$$

and then add this positive value Δx^+,

$$\frac{1}{\sqrt{N^d}} \sum_{\delta=0}^{N-1} e^{2\pi i \left(\frac{1}{m}\right)[\delta_1 (\nabla f)_1 + \delta_2 (\nabla f)_2 + \cdots + \delta_d (\nabla f)_d]} |\delta\rangle \tag{3.12}$$

3.1.2 Quantum gradient descent algorithm

Based on the equations derived in the previous sections, we now present the quantum gradient descent algorithm.

Input :

$f(x)$: A real-valued objective function for which the gradient is to be computed (e.g., a loss function in neural networks).
m : A scaling factor estimating the order of magnitude of the function values, used to normalize phase encoding.
n : The number of qubits used to encode each variable. It is chosen such that $N = 2^n$ discretization levels are sufficient to represent the variable with desired precision.
N : The discretization level per variable, defined as $N = 2^n$.
d : The number of variables or features in the input vector x.

l : A small positive constant that controls the learning rate.
Δx : A small perturbation used in the finite difference method to estimate partial derivatives.
$x_1, x_2, \ldots, x_n$: Input data points to be encoded into quantum states.

Register initialization :

Prepare d input registers, each consisting of n qubits, denoted as $|x_1\rangle, |x_2\rangle, \ldots, |x_d\rangle$. Each qubit is initialized to the basis state $|0\rangle$.

Procedure :

1. **Feature encoding :** Apply a quantum feature map to the input data to encode classical features into quantum states $|\psi(x)\rangle$. This is typically done using parameterized single-qubit rotation gates such as $RX(\theta)$ and $RZ(\theta)$, along with entangling gates like CNOT to capture quantum correlations.
2. **Quantum initialization :** Apply a Hadamard gate to each qubit in the input register to create a uniform superposition. This operation transforms the initial quantum state into an equal superposition overall possible computational basis states, enabling parallel processing across configurations.

The transformation is described as :

$$|0\rangle^{\otimes n} \longrightarrow \frac{1}{\sqrt{N^d}} \sum_{\delta=0}^{N^d-1} |\delta\rangle \tag{3.13}$$

Here, this qubit $|0\rangle^{\otimes n}$ denotes the initial quantum state of a register consisting of n qubits, where each qubit is individually initialized to the basis state $|0\rangle$, forming the tensor product.

In this equation :

n : Number of qubits required to encode the input data.
d : Number of input variables (features).
N^d : Total number of possible input combinations across all features.
$|\delta\rangle$: Computational basis state corresponding to a specific configuration.

This initialization is fundamental to achieving quantum parallelism, enabling the quantum system to simultaneously represent and process all potential input states.

$$|0\rangle^{\otimes n} \rightarrow \frac{1}{\sqrt{N^d}} \sum |\delta\rangle. \tag{3.14}$$

Register Initialization :

Apply the Quantum Addition (QADD) operator to add Δx to the superposition state :

$$\frac{1}{\sqrt{N^d}} \sum |\delta\rangle \xrightarrow{\text{QADD}} \frac{1}{\sqrt{N^d}} \sum |\delta + \Delta x\rangle, \tag{3.15}$$

Implement an oracle that computes the objective function f(x) and encodes its value into the phase of the quantum states :

$$O_f : \frac{1}{\sqrt{N^d}} \sum |\delta + \Delta x\rangle \rightarrow \frac{1}{\sqrt{N^d}} \sum e^{2\pi i \frac{N}{m} f(x)} |\delta + \Delta x\rangle. \tag{3.16}$$

Quantum Subtraction (QSUB) :

To facilitate gradient extraction, the quantum subtractor (QSUB) operation is applied to reverse the prior translation by Δx. This operation shifts the basis state $|\delta + \Delta x\rangle$ back to its original state $|\delta\rangle$, while preserving the phase information.

$$\frac{1}{\sqrt{N^d}} \sum_{\delta} e^{2\pi i \frac{N}{m} f(x)} |\delta + \Delta x\rangle \xrightarrow{\text{QSUB}} \frac{1}{\sqrt{N^d}} \sum_{\delta} e^{2\pi i \frac{N}{m} f(x)} |\delta\rangle. \tag{3.17}$$

Inverse quantum fourier transform (IQFT) :

Apply the Inverse Quantum Fourier Transform to transform the quantum state back into the computational basis :

$$\text{IQFT} : \frac{1}{\sqrt{Nd}} \sum e^{2\pi i \frac{N}{m} f(x)} |\delta\rangle \rightarrow \text{Fourier basis representing the gradients.} \tag{3.18}$$

Measurement :

Measure the quantum state after applying IQFT to extract the gradient values ∇f for the given objective function. The results will reflect the gradients in the computational basis :

$$\langle \nabla f(x) \rangle = \left(\frac{N}{m} \nabla f(\Delta x)\right)_1, \left(\frac{N}{m} \nabla f(\Delta x)\right)_2, \ldots, \left(\frac{N}{m} \nabla f(\Delta x)\right)_d \tag{3.19}$$

QEANN-QGD circuit

The Figure 3.3 is illustrates the quantum circuit that has been used in this project.

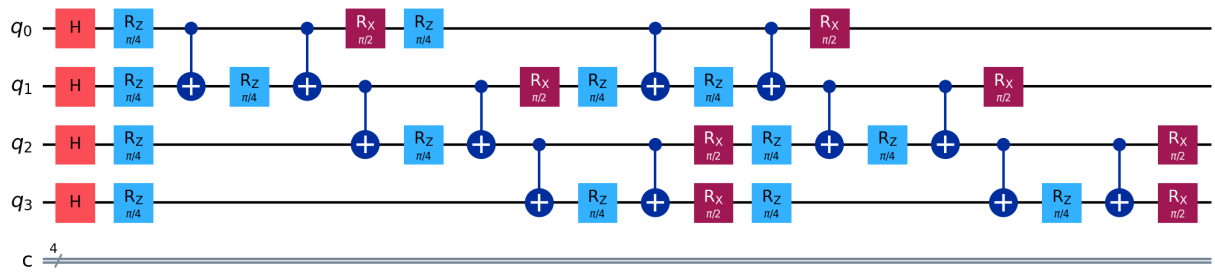


Figure 3.3 Quantum circuit

In the proposed quantum circuit, whichforms the quantum layer of the Quantum-Enhanced Adaptive Neural Network with Quantum Gradient Descent (QEANN-QGD), the design is carefully constructed to harness the power of quantum computation for feature extraction and optimization. The circuit comprises several key components that are essential for exploiting the advantages of quantum mechanics, such as superposition and entanglement, in the context of neural network training.This section is inspired by [20].

1. **Initialization with hadamard gates**

   At the beginning of the circuit, Hadamard gates are applied to each qubit (q_0, q_1, q_2, and q_3). The Hadamard gate serves to place the qubits into a superposition of states, allowing them to exist in both $|0\rangle$ and $|1\rangle$ simultaneously.

   The introduction of superposition at this stage of the circuit expands the capacity of the model to represent a richer set of feature spaces. In classical neural networks, feature representation is often limited by linear or non-linear transformations. In contrast, the superposition created by the Hadamard gates allows the quantum circuit to operate over a more complex space, which can lead to enhanced feature extraction capabilities.

   Thus, the Hadamard gates are positioned at the beginning of the circuit to set up the qubits for the subsequent quantum operations, ensuring that the circuit takes fulladvantage of quantum parallelism from the outset. The Hadamard gate is defined as[20]

   When applied to a single qubit in the basis state $|0\rangle$ or $|1\rangle$, the Hadamard transformation gives the superposition states

   For a four-qubit system, applying Hadamard gates to each qubit results in the tensor product state

$$H^{\otimes 4}|0000\rangle = \frac{1}{\sqrt{2^4}} \sum_{i=0}^{15} |i\rangle, \tag{3.20}$$

which represents an equal superposition of all possible computational basis states in a four-qubit quantum system.

2. **RZ and RX rotation gates**

After the qubits have been placed in superposition, a sequence of RZ and RX rotation gates is applied to each qubit. These gates are parameterized, meaning their rotation angles are learned during the training process. This allows the quantum circuit to adapt dynamically to the data, enhancing its ability to extract meaningful quantum features.

These parameterized rotations, when combined with the superposition introduced by the Hadamard gates, provide the model with the flexibility to explore a vast range of quantum state representations. By tuning the rotation angles, the quantum circuit can capture more complex relationships in data, enabling it to represent intricate structures that would be difficult to learn using classical methods alone.

Generating Entanglement with CNOT

Entanglement occurs when qubits become inseparably correlated, meaning that measurement of one qubit determines the state of the other, regardless of distance. The most common way to generate an entangled state using a CNOT gate is to apply it to a qubit in superposition [20].

To illustrate this, we consider a two-qubit system where the first qubit is placed in superposition using a Hadamard gate,

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle). \tag{3.21}$$

Applying a CNOT gate with the first qubit as the control, and the second qubit as the target, creates the well-known Bell state,

$$\text{CNOT} \quad \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle \ . \tag{3.22}$$

which is an entangled state where measuring one qubit immediately determines the state of the other.

Entanglement is essential in quantum neural networks because it enables the circuit to

model non-local dependencies between features in the input data. In classical neural networks, such relationships often require deep architectures or additional feature engineering to capture effectively. In contrast, quantum entanglement allows information to be distributed across qubits in a compact and efficient manner.

In our circuit, entanglement is introduced after the initial rotations to ensure that the learned data representations are shared across qubits in a meaningful way. This step is crucial because :

- It allows the quantum circuit to learn complex correlations between input features that classical models struggle to capture efficiently ;
- It enables the model to leverage quantum mechanics to represent higher-dimensional feature spaces ;
- It improves expressivity, allowing the circuit to approximate intricate patterns in data with fewer computational resources, when compared to classicalmethods.

By incorporating entanglement, the quantum-enhanced model can represent richer data structures, ultimately leading to improved learning performance in quantum machine learning tasks.

3. **Configuration of gates in the network**

The circuit begins with Hadamard gates on all qubits (q_0 to q_3) to place them into superposition, enabling quantum parallelism. This is followed by $R_z(\pi/4)$ gates applied to each qubit to introduce initial phase rotations as part of parameter encoding. A layer of CNOT gates is then used to entangle adjacentqubits, allowing quantum correlations to be captured. Subsequent layers alternate between $R_x(\pi/2)$ and $R_z(\pi/4)$ gates, which serve as trainable variational parameters : R_x gates rotate the state around the x-axis, while R_z gates adjust the phase. These rotation gates are interleaved with additional CNOT layers to repeatedly entangle qubits and distribute learned parameters across the system. This structure enhances the circuit's expressive power and its ability to model complex quantum states. The final layer of R_z and R_x gates fine-tunes the qubit states before measurement. Overall, the gate layout follows a typical variational pattern of initialization, entanglement, rotation, and readout preparation.

4. **Measurement**

The final stage of the quantum circuit involves measurement, which is the process of extracting classical information from quantum states. Measurement plays a crucial role in hybrid quantum-classical models such as QEANN-QGD, as it enables the transition from quantum computation to classical processing.

In quantum mechanics, measurement collapses a quantum state into one of the possible basis states, with probabilities determined by the state amplitudes. For a single qubit in a general superposition state,

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{3.23}$$

the probability of measuring the state as $|0\rangle$ or $|1\rangle$ is given by

$$P(0) = |\alpha|^2, \quad P(1) = |\beta|^2. \tag{3.24}$$

For an n-qubit system in the general quantum state,

$$|\Psi\rangle = \sum_{i=0}^{2^n-1} c_i|i\rangle, \tag{3.25}$$

the probability of collapsing to a specific classical outcome $|i\rangle$ is

$$P(i) = |c_i|^2, \tag{3.26}$$

where c_i represents the amplitude of the computational basis state $|i\rangle$.

In QEANN-QGD, measurement extracts quantum-enhanced features from the qubits, providing classical representations that serve as input to the fully connected layers of the neural network. The expectation value of a quantum observable $\hat{O}$ is computed as

$$\langle\hat{O}\rangle = \langle\Psi|\hat{O}|\Psi\rangle. \tag{3.27}$$

A common choice for the observable is the Pauli-Z operator, which provides measurement outcomes in the computational basis,

$$\langle Z\rangle = \langle\Psi|Z|\Psi\rangle = P(0) - P(1), \tag{3.28}$$

where Z is the Pauli-Z matrix

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{3.29}$$

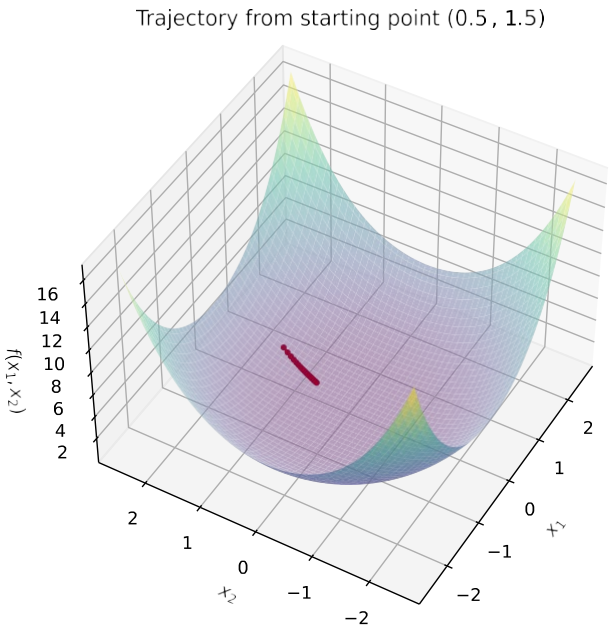
By measuring the expectation values of such observables, the quantum circuit generates

outputs that encode meaningful quantum correlations, which are then passed into the classical deep learning pipeline.

Measurement is a critical step in hybrid quantum-classical models because it serves as the bridge between quantum and classical computation. While quantum circuits leverage the principles of superposition and entanglement to perform complex transformations, their output exists in a probabilistic quantum state that cannot be directly accessed or interpreted by classical systems. Measurement collapses these quantum states into definite classical outcomes, thereby making the quantum information accessible for subsequent classical processing. This process is essential not only for extracting usable data from quantum systems but also for maintaining coherence between the quantum front-end and the classical back-end of the model. The results of measurement are employed in various roles within the learning pipeline, including :

- **Feature extraction** : Quantum measurement collapses the entangled and superposed quantum state into a classical probability distribution. These probabilities, derived from repeated measurements, encapsulate correlations and patterns learned by the quantum system. They are used as input features for classical components such as neural networks, capturing non-trivial transformations that are hard to reproduce classically ;
- **Decision making** : In classification or regression tasks, the outcomes of quantum measurements are post-processed by classical functions to determine the final prediction. For example, the most frequently measured basis state could represent the class label in a quantum classifier. This ensures that the probabilistic nature of quantum outputs is converted into deterministic decisions required in practical applications;
- **Further classical processing** : The extracted classical data from measurement are often fed into classical layers, such as dense or fully connected neural network layers. These layers continue the learning and decision-making process by refining the quantum features. This integration facilitates end-to-end learning across quantum and classical domains, allowing the model to benefit from quantum speedups and classical scalability simultaneously ;
- **Parameter updates** : Measurement outcomes also guide classical optimizers (e.g., gradient descent) in updating quantum circuit parameters. For instance, the measured expectation values are used to compute loss functions, which in turn influence the next iteration of quantum gate parameters through hybrid optimization loops;

Figures 3.4, 3.5 show how quantum gradient descent resembles the loss function.



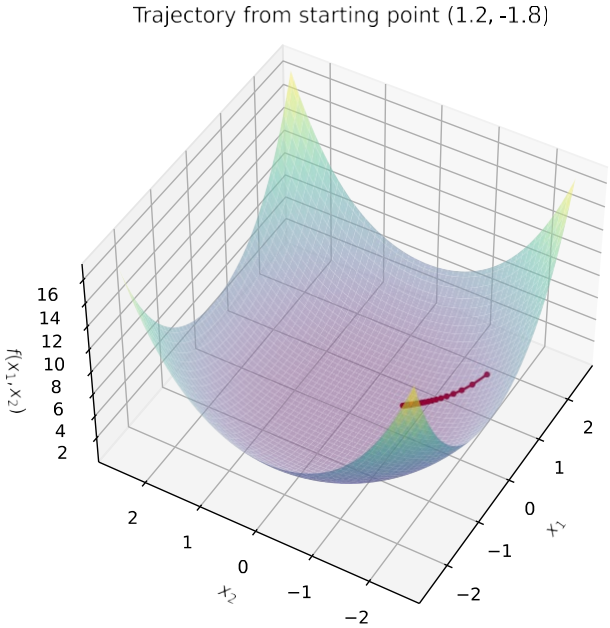
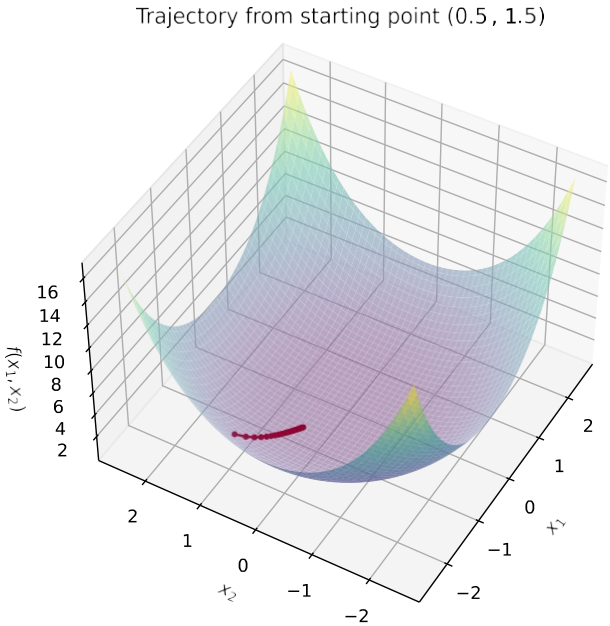


Figure 3.4 Top : Loss function trajectory starting from (0.5, 1.5). Bottom : Loss function trajectory starting from (1.2, -1.8).



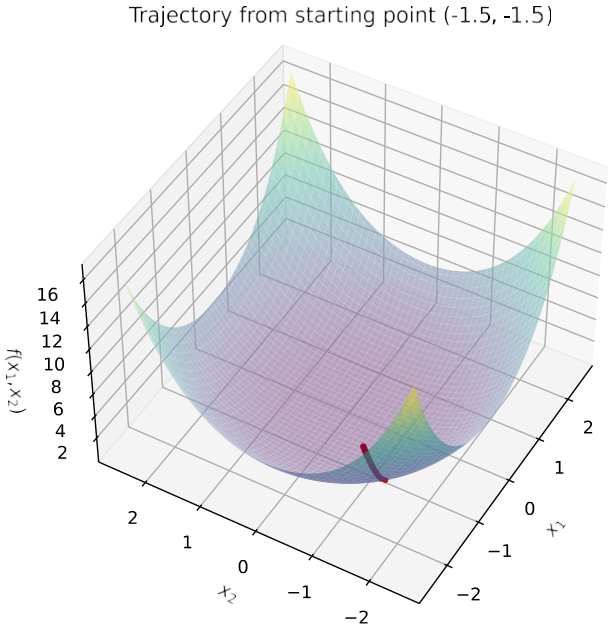


Figure 3.5 Top : Loss function trajectory starting from (-1.8, 1.0). Bottom : Loss function trajectory starting from (-1.5, -1.5).

3.2 Hybrid Feature Integration (HFI)

Hybrid Feature Integration (HFI) is a technique designed to combine quantum-enhanced and classical features within the QEANN-QGD framework, leveraging the strengths of both paradigms for improved feature extraction and pattern recognition. In deep learning, classical models, such as Convolutional Neural Networks (CNNs), excel at identifying local patterns like edges and textures. However, they struggle to efficiently capture complex, global dependencies. Theoretically, quantum computing has the potential to provide advantages through high-dimensional feature representations and entanglement, which could allow it to encode and process intricate relationships in data. HFI aims to integrate these capabilities, enhancing the overall learning performance of neural networks by leveraging both classical and quantum approaches.

The term Hybrid Feature Integration reflects the core idea of this method : integrating two distinct feature sets—one extracted through classicaldeep learning techniques and the other derived from quantum computations—into a unified representation. Rather than relying only on classical or quantum methods, HFI exploits their complementary strengths. The quantum component provides an alternative feature space that encodes data relationships differently than classical models, while CNNs effectively capture structured patterns. The fusion of these feature representations allows the model to benefit from both local and global data dependencies.

HFI is structured into three key stages, First feature extraction is the stage that involves extracting features separately from quantum and classical components. In the quantum pathway, classical input data is mapped to a quantum state with quantum feature map. This encoding leverages quantum properties such as superposition and entanglement. The Parameterized Quantum Circuit (PQC) then applies trainable quantum gates, such as controlled-X (CX) gates, to transform these quantum states into meaningful quantum-enhanced features. The resulting quantum features are subsequently measured and converted into classical values.

Simultaneously, the classicalcomponent of the model extracts features using Convolutional Neural Networks (CNNs). This process includes applying convolutional layers to detect fundamental patterns such as edges, textures, and shapes, followed by activation functions like ReLU to introduce non-linearity. Max pooling is also used to downsample the data while preserving essential information. These classical operations extract structured, localized features that complement the quantum-derived information.

Second, once quantum and classical features have been extracted, feature integration is per-

formed by concatenating their respective feature vectors. This is achieved by flattening the quantum output state into a vector and concatenating it with the classical CNN feature map. The combined vector is then passed through fully connected layers to learn a joint representation. This integration enables the model to leverage both the global entanglement captured by quantum circuits and the localized patterns extracted by CNNs, thus enhancing its ability to recognize complex data structures.

Third ; feature processing and learning that integrated feature vector is then passed through fully connected layers, where further transformations occur to optimize classification performance. These layers help refine the learned representations and adapt them to the final task, such as image classification or pattern recognition.

In the quantum pathway, classical input data is mapped to a quantum state with quantum feature map. This transformation enables the circuit to operate in a higher-dimensional Hilbert space, allowing quantum-enhanced feature representations.

One common method for encoding features is amplitude encoding, which maps classical data into the amplitude of quantum states :

$$|\psi(X_q)\rangle = \sum_{i=1}^{N} x_i |i\rangle, \quad \text{where} \quad \sum_{i=1}^{N} |x_i|^2 = 1. \tag{3.30}$$

Alternatively, angle encoding can be used, where each classical feature x_i is mapped to the rotation angle of a quantum gate applied to the i-th qubit. The resulting quantum state is :

$$|\psi(X_q)\rangle = \bigotimes_{i=1}^{n} R_Y(x_i)|0\rangle, \tag{3.31}$$

where $R_Y(x_i)$ is the rotation gate around the Y-axis by angle x_i, and the tensor product ($\bigotimes_{i=1}^{n}$) indicates that this operation is applied independently to each of the n qubits initialized to the state $|0\rangle$.

$$R_Y(\theta) = \begin{bmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{bmatrix}. \tag{3.32}$$

These encoding schemes allow quantum circuits to process data in ways that differ from classical methods.

The Parameterized Quantum Circuit (PQC) applies trainable quantum gates to transform the quantum state into meaningful quantum-enhanced features. A layer-wise transformation in the quantum circuit can be described as

$$\psi^{(l)} = U_{entangle} U(\theta^{(l)}) \psi^{(l-1)}, \tag{3.33}$$

where $U(\theta^{(l)})$ representstrainablerotation gatesthatadjustfeatureencoding; $U_{entangle}$ applies entangling operations between qubits, and $\psi^{(l)}$ is the quantum state at layer l.

Since entanglement enhances feature expressivity, we define the entanglement unitary as

$$U_{entangle} = \prod_{(i,j)\in E} CNOT(q_i, q_j). \tag{3.34}$$

where E represents the set of entangled qubit pairs. Once quantum and classical features have been extracted, they are merged into a unified representation. This fusion process is mathematically expressed as

$$F_{HFI} = Concat(F_{classical}, F_{quantum}), \tag{3.35}$$

where $F_{classical}$ is the feature vector extracted by convolutional neural networks (CNNs). $F_{quantum}$ is obtained from quantum circuit measurements. Since quantum measurements provide expectation values, we can represent quantum feature extraction as

$$F_{quantum} = \langle \psi | Z | \psi \rangle, \tag{3.36}$$

where Z is the Pauli-Z observable, measuring the probability of each qubit being in the $|0\rangle$ or $|1\rangle$ state. After the quantum-classical feature fusion, the resulting vector undergoes transformations in fully connected layers ;

$$\hat{y} = \sigma(W F_{HFI} + b), \tag{3.37}$$

where $\hat{y}$ is the model's prediction. W is the trainable weight matrix, and b is the bias vector. σ is an activation function. For multi-class classification, the final layer applies the softmax activation function [35].

$$P(y_i | F_{HFI}) = \frac{e^{W_i F_{HFI} + b_i}}{\sum_j e^{W_j F_{HFI} + b_j}}, \tag{3.38}$$

where $P(y_i)$ represents the probability of predicting class i.

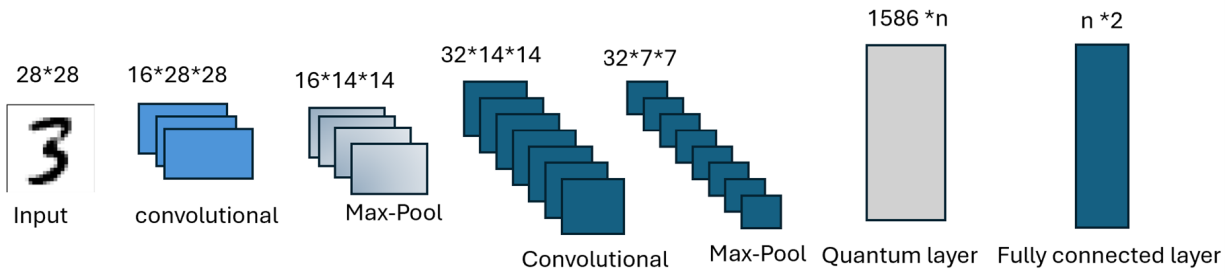


Figure 3.6 QEANN-QGD diagram

The primary advantage of the Hybrid Feature Integration (HFI) mechanism, as illustrated in Figure 3.6, lies in its ability to combine quantum-enhanced and classical features, resulting in more robust models. This integration is achieved by first extracting classicalfeaturesusing convolutional and pooling layers, followed by quantum feature extraction using parameterized quantum circuits. The resulting feature vectors from both domains are then concatenated into a single hybrid vector. This fused representation is subsequently passed through a fully connected layer for final classification. By integrating these two feature sets, HFI enables models to leverage both global quantum correlations and localized classical patterns. By integrating these two feature sets, HFI enables models to :

- **Capture complex data structures** : it effectively handles both local and global dependencies by leveraging classical feature extraction techniques alongside quantum-enhanced representations.
- **Use high-dimensional feature spaces** : it exploits the theoretically rich feature space provided by quantum circuits, allowing for more expressive data representations.
- **Enhance generalization** : it improves model adaptability by integrating complementary quantum and classical learning approaches.
- **Increase flexibility** : it allows models to be tailored to various datasets and tasks, making it a versatile approach for different machine learning applications.
- **Improve performance across multiple domains** : it facilitates better results in tasks such as image classification, speech recognition, and time-series prediction.

3.2.1 Integration with hybrid quantum-classical models

The title Integration with hybrid quantum-classical models is chosen to reflect the essential role of Quantum Principal Component Analysis (QPCA) in bridging quantum and classical computation. In hybrid models, quantum data must be efficiently processed alongside classical data, requiring effective techniques for feature extraction and dimensionality reduction.

QPCA facilitates this integration by reducing the complexity of quantum data while preserving its most significant features, ensuring compatibility with classical machine learning algorithms.

HybridFeatureIntegration(HFI) in QPCAplays acrucialrole in mergingquantum-extracted features with classical features. This combination enables a more comprehensive representation of data by leveraging the strengths of both quantum and classical computations. By reducing dimensionality, QPCA ensures that quantum-derived information can be efficiently incorporated into classical models, enhancing their ability to recognize patterns and extract meaningful insights.

QPCA can be applied to a wide range of machine learning tasks, including image classification, data clustering, and anomaly detection. The quantum-reduced data is seamlessly fed into classicalmachine learning models, such as neural networks or support vectormachines, to perform classification, decision-making, or other predictive tasks.

The advantages of QPCA in hybrid quantum-classical models include [16] :

- **Quantum-accelerated eigenvalue decomposition** : QPCA leverages quantum algorithms for eigenvalue decomposition, offering a significant speedup over classical PCA, particularly for high-dimensional datasets ;
- **Efficient dimensionality reduction** : QPCA reduces the complexity of quantum data while retaining the most relevant features, enabling faster and more accurate processing in hybrid models ;
- **Noise reduction** : By focusing on principal components, QPCA filters out irrelevant or noisy quantum features, improving the overall quality of the quantum data;
- **Seamless integration with classical algorithms** : The quantum-reduced data from QPCA can be effortlessly integrated with classical machine learning models, enabling the development of robust hybrid algorithms ;
- **Scalability for large datasets** : QPCA is well-suited for handling large datasets that would be computationally infeasible to process classically, providing a scalable solution for quantum machine learning applications.

In QPCA, classical data is first encoded into a quantum state as a density matrix,

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|, \tag{3.39}$$

where ρ represents the density matrix encoding the data, p_i are probability weights, and $|\psi_i\rangle$ are the quantum states corresponding to data points.

The goal of QPCA is to extract the dominant eigenvectors of ρ, which correspond to the most significant principal components of the dataset. QPCA estimates the eigenvalues λ_i and eigenvectors $|\psi_i\rangle$ of the density matrix using quantum phase estimation (QPE) :

$$\rho|\psi_i\rangle = \lambda_i|\psi_i\rangle, \tag{3.40}$$

where λ_i represents the significance of the principal component $|\psi_i\rangle$. Quantum phase estimation provides an efficient means of computing these eigenvalues, offering an exponential speedup over classical PCA for high-dimensional data [16].

Once the eigenvalues are computed, QPCA retains only the top-k eigenvectors that correspond to the largest eigenvalues,

$$\rho_k = \sum_{i=1}^{k} \lambda_i |\psi_i\rangle\langle\psi_i|, \tag{3.41}$$

where k is the number of principal components retained, ρ_k is the reduced representation of the dataset.This ensures that the most informative features are preserved while eliminating lower-variance (noisy) components.Since classical machine learning models require numerical feature representations, the reduced quantum state ρ_k is measured to obtain classical feature vectors :

$$F_{QPCA} = \langle\psi|Z|\psi\rangle, \tag{3.42}$$

where Z is the Pauli-Z observable, used to extract feature values from the quantum-reduced representation. Once QPCA has reduced the dimensionality of quantum data, the extracted features are merged with classical feature vectors,

$$F_{Hybrid} = \mathrm{Concat}(F_{classical}, F_{QPCA}), \tag{3.43}$$

where $F_{classical}$ represents features obtained from classical machine learning techniques (e.g., PCA, CNNs, or handcrafted features). F_{QPCA} represents quantum-enhanced features. This fusion enables hybrid models to leverage both quantum and classical representations, improving pattern recognition and classification accuracy. After integration, the hybrid feature vector undergoes transformation through a classical neural network :

$$\hat{y} = \sigma(W F_{Hybrid} + b), \tag{3.44}$$

where W is the trainable weight matrix, b is the bias term, σ is the activation function (e.g., softmax for multi-class classification). For multi-class prediction, softmax is applied

$$P(y_i|F_{Hybrid}) = \frac{e^{W_i F_{Hybrid} + b_i}}{\sum_j e^{W_j F_{Hybrid} + b_j}}. \tag{3.45}$$

This final step enables hybrid models to make predictions based on a quantum-enhanced and classically refined feature space.

3.3 The adaptive method in quantum-classical neural networks

Adaptive optimization plays a critical role in the training of hybrid quantum-classical neural networks, where both quantum and classical components contribute to learning from data. In contrast to static parameter schemes, adaptive methods iteratively update the trainable parameters of the quantum circuit based on gradient information obtained from the loss function. This facilitates effective learning and enhances the model's generalization capability.

In hybrid architectures, the quantum circuit is parameterized by a set of variables θ, which control the behavior of quantum gates such as R_X, R_Y, R_Z. These gates manipulate the quantum state, enabling the extraction of features from data mapped into high-dimensional Hilbert spaces. The number and structure of these gates depend on the circuit design, typically involving both local (single-qubit) and entangling (multi-qubit) operations.

At the beginning of training, the quantum parameters are initialized either randomly or via a heuristic designed to improve convergence. Let the initial parameters be denoted as :

$$\theta^{(0)} = \{\theta_1^{(0)}, \theta_2^{(0)}, \ldots, \theta_n^{(0)}\}. \tag{3.46}$$

These parameters determine how the quantum circuit will process encoded classical data, influencing the evolution of the quantum state throughout the circuit.

The forward pass begins by encoding the classical input x_i into a quantum state using a feature map $U_{encode}(x_i)$, which maps data to a high-dimensional quantum Hilbert space :

$$|\psi(x_i)\rangle = U_{encode}(x_i)|0\rangle^{\otimes n}. \tag{3.47}$$

Subsequently, a parameterized quantum circuit $U(\theta)$ is applied to the encoded state :

$$|\psi_{out}\rangle = U(\theta)|\psi(x_i)\rangle, \tag{3.48}$$

where the circuit is defined as a sequence of unitary gates :

$$U(\theta) = \prod_{l} U_l(\theta_l), \tag{3.49}$$

with each $U_l(\theta_l)$ representing a trainable quantum gate at layer l. The final quantum state is measured to extract an expectation value with respect to a Hermitian operator (typically the Pauli-Z observable) :

$$F_{quantum} = \langle \psi_{out} | Z | \psi_{out} \rangle, \tag{3.50}$$

which serves as input to the classical component of the hybrid model for loss computation. AS like as forward path, for the backward path we need to use parameter shift rule that will cover in the next section.

3.3.1 Parameter update using gradients

In the adaptive method, after computing the loss function, the next step involves calculating the gradients of the loss with respect to the quantum circuit parameters. In classical neural networks, this process is straightforward using automatic differentiation libraries, such as those in TensorFlow. However, in quantum circuits, gradient computation is more complex due to the probabilistic nature of quantum measurements, and the restrictions imposed by quantum mechanics.

Unlike classical backpropagation, where gradients are obtained by computing derivatives of the loss function analytically or numerically, quantum circuits require specialized techniques to estimate gradients. One of the most widely usedmethods for this purpose is the parameter shift rule. This rule allows gradients to be computed efficiently by evaluating the quantum circuit at slightly shifted parameter values [36]. Consider a parameterized quantum circuit where a unitary gate is defined as : $U(\theta)$ is a unitary gate parameterized by θ, representing a quantum operation. G is a Hermitian generator operator with eigenvalues $\pm 1/2$, driving the unitary evolution. θ is a real-valued parameter that controls the rotation angle in the unitary gate. ψ is a quantum state vector in the Hilbert space. O is a Hermitian observable operator whose expectation value is measured. $f(\theta)$ is the expectation value of O in the state transformed by $U(\theta)$, defined as $\langle \psi | U^{\dagger}(\theta) O U(\theta) | \psi \rangle$.

Derivation

A unitary gate is defined as :

$$U(\theta) = e^{-i\theta G}, \tag{3.51}$$

where G is a Hermitian operator with eigenvalues $\pm 1/2$, and θ is the parameter. The expectation value of an observable O in a state ψ is :

$$f(\theta) = \langle\psi|U^{\dagger}(\theta)OU(\theta)|\psi\rangle. \tag{3.52}$$

The goal is to compute the derivative $\frac{\partial f(\theta)}{\partial\theta}$.

The derivative of the expectation value is :

$$\frac{\partial f(\theta)}{\partial\theta} = \frac{\partial}{\partial\theta}\langle\psi|U^{\dagger}(\theta)OU(\theta)|\psi\rangle. \tag{3.53}$$

Using the product rule and the fact that $U^{\dagger}(\theta) = e^{i\theta G}$, we compute :

$$\frac{\partial U(\theta)}{\partial\theta} = \frac{\partial}{\partial\theta} e^{-i\theta G} = -iGe^{-i\theta G} = -iGU(\theta), \tag{3.54}$$

and similarly,

$$\frac{\partial U^{\dagger}(\theta)}{\partial\theta} = \frac{\partial}{\partial\theta} e^{i\theta G} = iGe^{i\theta G} = iGU^{\dagger}(\theta). \tag{3.55}$$

Thus, the derivative becomes :

$$\frac{\partial f(\theta)}{\partial\theta} = \langle\psi| \frac{\partial U^{\dagger}(\theta)}{\partial\theta} OU(\theta) + U^{\dagger}(\theta)O\frac{\partial U(\theta)}{\partial\theta} |\psi\rangle. \tag{3.56}$$

$$\frac{\partial f(\theta)}{\partial\theta} = \langle\psi| \left(iGU^{\dagger}(\theta)OU(\theta) - iU^{\dagger}(\theta)OGU(\theta) \right) |\psi\rangle = i\langle\psi|[G, U^{\dagger}(\theta)OU(\theta)]|\psi\rangle, \tag{3.57}$$

To derive the parameter shift rule, consider the expectation value at shifted parameters. Since G has eigenvalues $\pm 1/2$, we can use the spectral properties of G. The unitary at a shifted parameter is :

$$U(\theta + s) = e^{-i(\theta+s)G} = e^{-i\theta G}e^{-isG} = U(\theta)e^{-isG}. \tag{3.58}$$

The expectation value at $\theta + s$ is :

$$f(\theta + s) = \langle\psi|U^{\dagger}(\theta + s)OU(\theta + s)|\psi\rangle = \langle\psi|e^{isG}U^{\dagger}(\theta)OU(\theta)e^{-isG}|\psi\rangle. \tag{3.59}$$

Define the shifted operator $O(\theta) = U^\dagger(\theta) O U(\theta)$. Then :

$$f(\theta + s) = \langle\psi| e^{isG} O(\theta) e^{-isG} |\psi\rangle. \tag{3.60}$$

For G with eigenvalues $\pm 1/2$, since $G^2 = \frac{1}{4} I$ (because the eigenvalues squared are $(1/2)^2 = 1/4$), we express e^{-isG} using the spectral decomposition :

$$e^{-isG} = \cos\left(\frac{s}{2}\right) I - 2i \sin\left(\frac{s}{2}\right) G, \tag{3.61}$$

and similarly :

$$e^{isG} = \cos\left(\frac{s}{2}\right) I + 2i \sin\left(\frac{s}{2}\right) G. \tag{3.62}$$

Thus, the expectation value becomes :

$$f(\theta + s) = \langle\psi| \cos\left(\frac{s}{2}\right) I + 2i \sin\left(\frac{s}{2}\right) G \; O(\theta) \cos\left(\frac{s}{2}\right) I - 2i \sin\left(\frac{s}{2}\right) G \; |\psi\rangle. \tag{3.63}$$

Expanding and simplifying, we focus on terms relevant to the derivative. Evaluate at specific shifts $s = \pm\pi/2$:

$$f\left(\theta + \frac{\pi}{2}\right) = \langle\psi| e^{i\frac{\pi}{4}G} O(\theta) e^{-i\frac{\pi}{4}G} |\psi\rangle, \quad f\left(\theta - \frac{\pi}{2}\right) = \langle\psi| e^{-\frac{\pi}{4}G} O(\theta) e^{i\frac{\pi}{4}G} |\psi\rangle. \tag{3.64}$$

Consider the difference :

$$f\left(\theta + \frac{\pi}{2}\right) - f\left(\theta - \frac{\pi}{2}\right) = \langle\psi| \; e^{i\frac{\pi}{4}G} O(\theta) e^{-i\frac{\pi}{4}G} - e^{-i\frac{\pi}{4}G} O(\theta) e^{i\frac{\pi}{4}G} \; |\psi\rangle. \tag{3.65}$$

With $s = \pi/2$, we get $\cos(\pi/4) = \frac{\sqrt{2}}{2}$, $\sin(\pi/4) = \frac{\sqrt{2}}{2}$. Relating this to the commutator in Equation (3.59), the difference is proportional to the commutator term :

$$f\left(\theta + \frac{\pi}{2}\right) - f\left(\theta - \frac{\pi}{2}\right) = 2i\langle\psi|[G, O(\theta)]|\psi\rangle. \tag{3.66}$$

We obtain :

$$\frac{\partial f(\theta)}{\partial\theta} = \frac{1}{2} \; f\left(\theta + \frac{\pi}{2}\right) - f\left(\theta - \frac{\pi}{2}\right) \; . \tag{3.67}$$

The parameter shift rule enables the use of gradient-based optimization techniques in quantum models, making it possible to implement training algorithms such as Quantum Gradient Descent (QGD). These methods adjust the quantum parameters iteratively to minimize the loss function, ensuring that the quantum model learns effectively from the data.

Once the gradients have been computed, the next step is the adaptive quantum parameter

optimization, where quantum parameters θ are updated using an optimization algorithm. The choice of optimizer depends on the specific application and dataset characteristics. Common optimization methods include :

1. **Stochastic gradient descent (SGD) [37] :** basic yet effective optimization method that updates the parameters in the direction of the negative gradient, following the rule,

$$\theta_{t+1} = \theta_t - \eta \frac{\partial L}{_t},$$

   where η is the learning rate, θ_t represents the parameter at iteration t, and L is the loss function.

2. **Adam optimizer [38] :** an adaptive optimization method that combines the benefits of momentum-based updates. Adam dynamically adjusts the learning rate for each parameter, making it well-suited for training deep quantum models,

$$\theta_{t+1} = \theta_t - \frac{\eta m_t}{\sqrt{v_t} + \epsilon}, \tag{3.69}$$

   where m_t and v_t are estimates of the first and second moments of the gradients, respectively, and ϵ is a small constant for numerical stability.

The adaptive optimization process is repeated for each training epoch, iteratively updating the quantum parameters based on feedback from the loss function. By continuously refining the quantum circuit parameters, the model progressively improves its performance, ensuring better convergence and more effective integration between quantum and classical learning components.

3.4 Noise implementation in quantum-enhanced adaptive neural networks (QEANN-QGD)

One of the most critical challenges in quantum machine learning is mitigating the effects of noise in quantum computations. Quantum noise is an inherent and unavoidable phenomenon that arises due to the fragility of quantum states, particularly when qubits interact with their environment. In our network, integrating effective noise mitigation techniques is essential for maintaining model accuracy and stability. This section explores the role of noise in quantum-enhanced learning and the strategies employed in our implementation to counteract its effects.

Quantum computing operates on qubits, which leverage quantum superposition to perform parallel computations. However, this same property makes qubits highly susceptible to noise. The main sources of quantum noise include :

- **Decoherence** [39] : the loss of quantum coherence, causing qubits to collapse prematurely into classical states ;
- **Gate errors** [7] : imperfections in the implementation of quantum gates introduce inaccuracies in qubit transformations ;
- **Environmental interactions** [40] : external temperature fluctuations and electromagnetic interference can cause qubit states to decay, particularly affecting superconducting qubits ;
- **Measurement errors** [41] : Errors introduced during the final readout of quantum states, leading to incorrect classical outputs.

In QEANN-QGD, the model incorporates key quantum machine learning methods such as quantum principal component analysis (QPCA), Variational quantum eigensolver (VQE), and quantum gradient descent (QGD). Each of these techniques depends on accurate quantum feature extraction, making noise management a crucial aspect of ensuring the integrity of learned representations.

Beyond depolarizing noise, additional noise models are considered to further enhance the robustness of QEANN-QGD. These include :

- **Amplitude damping** [42] : represents energy loss in qubits, simulating how quantum states decay over time due to environmental interactions.
- **Phase damping** [43] : models the loss of coherence in quantum states without energy dissipation, affecting interference-based quantum computations.

By incorporating these noise models, the quantum circuits in QEANN-QGD are designed to better reflect realistic quantum hardware conditions. This approach allows us to evaluate and mitigate noise-related performance degradation, ensuring that the extracted quantum-enhanced features remain stable and useful for hybrid quantum-classical learning.

Noise-aware quantum modeling is an essential component of quantum-enhanced machine learning. By simulating and addressing real-world noise effects, QEANN-QGD improves its ability to generalize effectively across different datasets and enhances the practical feasibility of hybrid quantum-classical neural networks.

3.5 Noise reduction techniques in QEANN-QGD

Implementing full-scale Quantum Error Correction (QEC) is impractical due to its high qubit overhead and computational complexity. Instead, the model uses a combination of practical noise reduction techniques that improve stability and reliability without requiring large-scale quantum resources. The following techniques are specifically chosen to address different types

of quantum noise while ensuring efficient quantum-classical integration.

Noise-resilient quantum gates

Quantum gates are fundamental components of quantum circuits, responsible for manipulating qubit states [44]. However, certain gate implementations are inherently more susceptible to noise, particularly in noisy intermediate-scale quantum (NISQ) devices. In QEANN-QGD, we optimize gate selection to improve noise resilience by implementing :

- **Clifford gates** [41] : gates such as the Hadamard (H), Pauli (X, Y, Z), and CNOT gates exhibit higher resilience to certain noise effectsand are commonly used in stabilizer circuits ;
- **Dynamically corrected gates** [41] : these gates use carefully structured pulse sequences to counteract decoherence effects and minimize systematic errors.

Noise-resilient gates are implemented by leveraging the backend noise models of quantum simulator. By selecting gates with lower intrinsic error rates and optimizing circuit structure, we improve the reliability of quantum computations without adding extra qubits.

Error mitigation with zero-noise extrapolation

Since full quantum error correction is not feasible, Zero-Noise Extrapolation (ZNE) is applied to reduce noise effects without additional qubits [45]. ZNE is a post-processing technique that estimates the result of a quantum computation in an ideal, noise-free scenario by systematically measuring the circuit at varying noise levels.

ZNE is implemented in three steps :

1. The same quantum circuit is executed multiple times with artificially increased noise (e.g., applying additional depolarizing noise gates) ;
2. The output results are analyzed to model how noise affects computation ;
3. A mathematical extrapolation is performed to estimate the noise-free result.

This approach allows QEANN-QGD to improve the accuracy of quantum feature extraction while maintaining computational feasibility on devices.

Hybrid noise management in classical layers

Even after applying quantum noise reduction techniques, some errors remain in the extracted quantum features. To further improve reliability, the classical neural network component of

QEANN-QGD incorporates hybrid noise management, which refines the quantum-derived data before further processing.

Hybrid noise management is achieved through :

- **Denoising filters** [46] : classical smoothing techniques (e.g., Gaussian filters) are applied to reduce fluctuations in quantum-extracted features caused by noise during measurement ;
- **Regularization techniques** [35] : methods such as dropout and L2 regularization prevent the classical neural network from overfitting to noisy quantum features ;
- **Data augmentation** : training data is artificially expanded with variations of quantum feature noise, allowing the model to generalize better to real-worldconditions.

By combining these noise reduction strategies, QEANN-QGD ensuresmore stable quantum computations while leveraging classical post-processing to improve learning efficiency. This integrated approach allows hybrid quantum-classical models to function effectively despite hardware-imposed noise limitations.

Parameterized quantum circuits and Ansatz in VQE

The success of the variational quantum eigensolver (VQE) algorithm heavily depends on the choice of the Ansatz, also known as the parameterized quantum circuit (PQC). The Ansatz defines the quantum state $|\psi(\theta)\rangle$, where θ represents the set of tunable parameters optimized to minimize the expectation value of the Hamiltonian,the Hamiltonian, whichrepresentsthe total energy operator of the quantum system and governs its evolution. The goal of VQE is to approximate the ground state energy of a quantum system by iteratively adjusting these parameters through a hybrid quantum-classical optimization loop.

In our implementation, the PQC used in the VQE step consists in layers of quantum gates such as RX (rotation along the X-axis), RZ (rotation along the Z-axis), and CX (controlled-NOT) gates. The structure and depth of the Ansatz directly impacts the expressiveness of the quantum circuit. A deeper circuit can represent more complex quantum states but introduces higher susceptibility to noise and requires more computational resources. Therefore, in QEANN-QGD, we use a relatively shallow Ansatz to strike a balance between expressiveness and practical feasibility, considering the constraints of current quantum hardware [47].

In the context of VQE, an Ansatz is a quantum circuit with tunable parameters designed to approximate the eigenstates of the Hamiltonian being studied. The quality of the approximation depends on the structure of the Ansatz and the ability of the classical optimizer to adjust the parameters effectively.

The choice of Ansatz is critical because :
- it determines the range of quantum states that the circuit can explore ;
- it affects the convergence speed of the optimization process ;
- it influences the resilience of the VQE algorithm to quantum noise.

In QEANN-QGD, the Ansatz must be carefully designed to maximize quantum feature extraction while remaining robust to hardware noise.

Quantum computers are inherently noisy, and the performance of the VQE is directly affected by gate errors, measurement errors, and decoherence. These noise sources introduce inaccuracies in the estimation of the ground state energy, making it essential to incorporate noise mitigation techniques.

Role of the classical optimizer in VQE

A key feature of the VQE is its hybrid nature, where quantum circuits are used to compute expectation values [48], while a classical optimizer updates the parameters θ. The classical optimization step plays a crucial role in mitigating noise effects by :
- adjusting parameters based on noisy quantum measurements, ensuring more stable convergence ;
- reducing the impact of local minima by adapting learning rates dynamically ;
- enhancing robustness by filtering out small fluctuations due to quantum errors.

In QEANN-QGD, we use Quantum Gradient Descent (QGD) to optimize the quantum parameters based on gradients of the loss function. QGD enables efficient parameter updates while maintaining adaptability to noisy environments.

Advantages of VQE in hybrid quantum-classical models

The use of VQE within hybrid quantum-classical frameworks like QEANN-QGD provides several key advantages [49] :

- **Efficient use of quantum resources :** VQE offloads parameter optimization to classical computing while using quantum circuits for the computationally expensive part (evaluating expectation values of the Hamiltonian) ;
- **Scalability :** the Ansatz depth and the number of qubits can be adjusted to scale VQE for larger quantum systems, making it adaptable to more complex quantum models ;
- **Adaptability :** VQE is flexible in its application ; by modifying the Hamiltonian and Ansatz, it can be tailored for different quantum problems ;
- **Robustness to Noise :** unlike fully quantum algorithms such as Quantum Phase Esti-

mation (QPE), VQE leverages classical optimization, making it more resilient to noise ;

- **Flexibility in optimization :** different classical optimizers can be used in VQE. In QEANN-QGD, we use Quantum Gradient Descent (QGD) to update the quantum parameters, enhancing adaptability and efficiency.

3.6 Network architecture and implementation

The network consist of 4 qubits, 4 CNN layers, 4 quantum layers and 2 fully connectedlayers. Also 70 percent of the data used for training and 30 percent for testing it.

In our network, multiple libraries are used, each serving distinct roles within the classical and quantum parts of the model. The following libraries are used in our implementation : TensorFlow, Keras, Qiskit, PennyLane, NumPy, Matplotlib, and Scikit-learn.

TensorFlow is used for building and training the classical deep learning components. Specifically, functions such as tensorflow.keras.models are used to define and compile neural network models, while Conv2D and Flatten layers from tensorflow.keras.layers are used to build convolutional and fully connected layers. The Model class is used to train and evaluate the neural network models. Adam and SGD optimizers are also imported from tensorflow.keras.optimizers to perform gradient-based optimization of the network parameters.

Keras APIs from TensorFlow are used to simplify the creation of models and layers, enabling the efficient implementation of complex architectures. The ease of defining layers like Conv2D for convolution operations and Dense for fully connected layers makes Keras suitable for integrating the classical components of the model.

Qiskit is used to build the quantum circuits and to perform quantum simulations. The QuantumCircuit function allows for defining quantum layers in which quantum gates like RX, RZ, and CX are applied to the qubits. These operations enable the quantum feature extraction component of the hybrid model. The quantum circuits are simulated using the Aer module from Qiskit, while measurement operations are used to extract quantum states for further processing in the hybrid model.

PennyLane is used to facilitate the integration between classical deep learning and quantum computing. The functions from PennyLane such as qml.QNode allow the definition of quantum nodes, which serve as quantum layers in the hybrid architecture. PennyLane also provides automatic differentiation through quantum differentiation techniques like the parameter_shift rule, which enables the calculation of gradients for the quantum parameters.

NumPy is used extensively for numerical operations, array manipulations, and matrix handling within both the classical and quantum parts of the model. Functions from Numpy are

employed to manage data transformations, pre-processing, and matrix calculations, which are fundamental for both defining quantum circuits and handling classicallayers.

Matplotlib is used to generate plots that track the model's performance across training epochs. Specifically, functions from matplotlib.pyplot are used to visualize metrics such as accuracy, loss, and ground state energy. These visualizations are critical for understanding the convergence and effectiveness of the quantum-classical hybrid model.

Scikit-learn is imported for its utility in data preprocessing and model evaluation. Functions from sklearn.metrics are used to compute evaluation metrics like precision, recall, F1-score, and ROC-AUC, which provide a comprehensive assessment of model performance. Additionally, train_test_split from sklearn.model_selection is used to divide the dataset into training and testing sets, ensuring proper validation during the model's training phase.

These libraries collectively allow for the construction, training, and evaluation of a hybrid quantum-classical neural network, facilitating integration between quantum circuits and classical neural networks, with extensive visualization and evaluation capabilities. The evaluation includes key performance metrics such as [50] :

- **Accuracy** : The ratio of correctly predicted samples to the total number of predictions made. It reflects the overall correctness of the model.
- **Loss** : A value that quantifies how well or poorly the model's prediction matches the actual outcome. Lower loss indicates better performance.
- **Precision** : The ratio of true positives to the total predicted positives. It measures the model's exactness—how many selected items are relevant.
- **Recall** : The ratio of true positives to the total actual positives. It evaluates the model's ability to find all relevant cases.
- **F1-score** : The harmonic mean of precision and recall. It balances the two, especially useful when classes are imbalanced.
- **ROC AUC (Receiver Operating Characteristic - Area Under Curve)** : A metric that shows the model's ability to distinguish between classes. A higher AUC indicates better classification performance.

CHAPTER 4 EXPERIMENTAL RESULTS

This chapter presents the experimental evaluation of classical and quantum optimization methods applied to image classification tasks. We compare the performance of classical optimizers stochastic gradient descent (SGD) and Adam with the proposed hybrid Quantum-Enhanced Adaptive Neural Network using Quantum Gradient Descent (QEANN-QGD). The models are tested on two benchmark datasets : MNIST and CIFAR-10.

The evaluation includes key performance metrics such as accuracy, loss, precision, recall, F1-score, and ROC AUC. Through this comparison, our objective is to assess both learning efficiency and final performance across optimization strategies.

4.1 SGD performance on the MNIST dataset

In this subsection, we focus on analyzing the behavior of the Stochastic Gradient Descent (SGD) optimizer when applied to the MNIST dataset. Figures 4.1 and 4.2 illustrate the training performance of the Stochastic Gradient Descent (SGD) optimizer on the MNIST dataset.

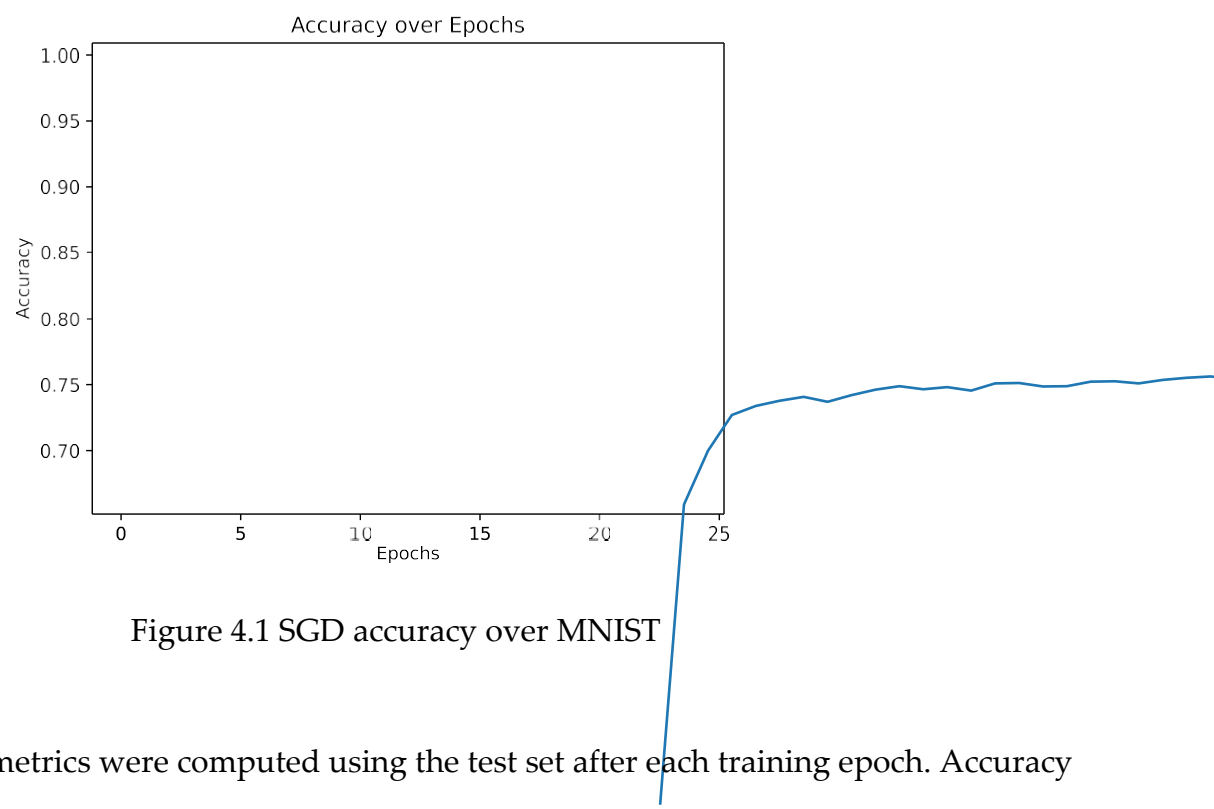


Figure 4.1 SGD accuracy over MNIST

The evaluation metrics were computed using the test set after each training epoch. Accuracy

was measured as the proportion of correctly classified images relative to the total number of test samples. The loss was calculated using the categorical cross-entropy function, which quantifies the difference between the predicted probability distribution and the true label distribution. Additionally, precision, recall, F1-score, and ROC AUC were computed, based on the predicted labels and the ground truth for the test set. These metrics provide a comprehensive assessment of model performance, particularly in terms of classification quality and robustness.

In Figure 4.1 illustrates, the model exhibits rapid accuracy gains within the first few epochs, reaching a peak accuracy of approximately 99.84% by epoch 5. The accuracy remains consistently high throughout the remainder of the training process, indicating that the model has successfully generalized over the dataset. This fast and stable convergence highlights the suitability of SGD for classification tasks such as MNIST.

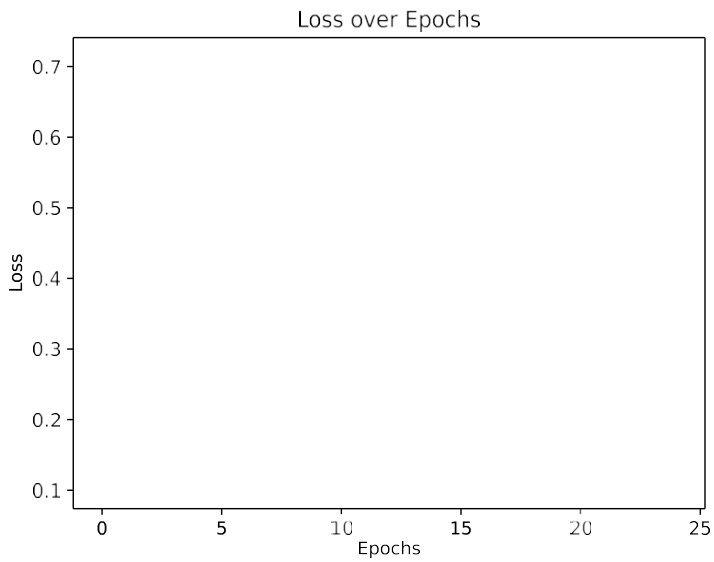


Figure 4.2 SGD loss over MNIST

Figure 4.2 illustrates the corresponding training loss, which starts at a relatively high value near 0.7 and decreases sharply during the early stages of training. By epoch 10, the rate of decline slows, and the loss gradually stabilizes around 0.1. Despite not reaching a lower loss value, this behavior suggests convergence and sufficient error minimization. This pattern suggests effective error minimization and convergence toward an optimal solution. The results confirm the effectiveness of SGD in minimizing prediction error on this dataset while maintaining training stability.

4.2 Adam method performance on the MNIST dataset

This section analyzes the performance of the Adam optimizer on the MNIST dataset, focusing on both accuracy and loss during training. Adam is known for its adaptive learning rate mechanism, which adjusts the step size for each parameter, making it suitable for training deep learning models efficiently. The results presented below demonstrate how Adam performs in terms of convergence speed, prediction accuracy, and loss reduction when applied to image classification task. Figures 4.3 and 4.4illustrate the training accuracy and loss curves of the Adam optimizer on the MNIST dataset.

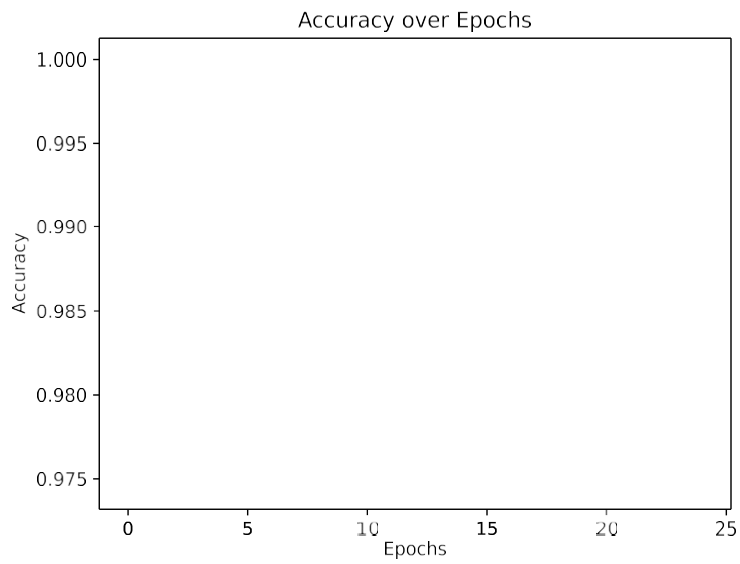


Figure 4.3 Accuracy of Adam over MNIST

In Figure 4.3, the model starts with an accuracy near 97% and rapidly improves within the first few epochs, reaching approximately 99.97%. After this initial rise, the accuracy remains consistently high with minor fluctuations, stabilizing just below 100%. This demonstrates Adam's ability to converge quickly and maintain high classification performance. The minimal variance suggests that Adam's adaptive learning rate strategy effectively fine-tunes model weights during training. The steady performance across epochs indicates that the model is not suffering from significant overfitting. This level of accuracy reflects strong generalization capability on the MNIST dataset. Overall, Adam proves to be a reliable optimizer for image classification tasks involving different datasets.

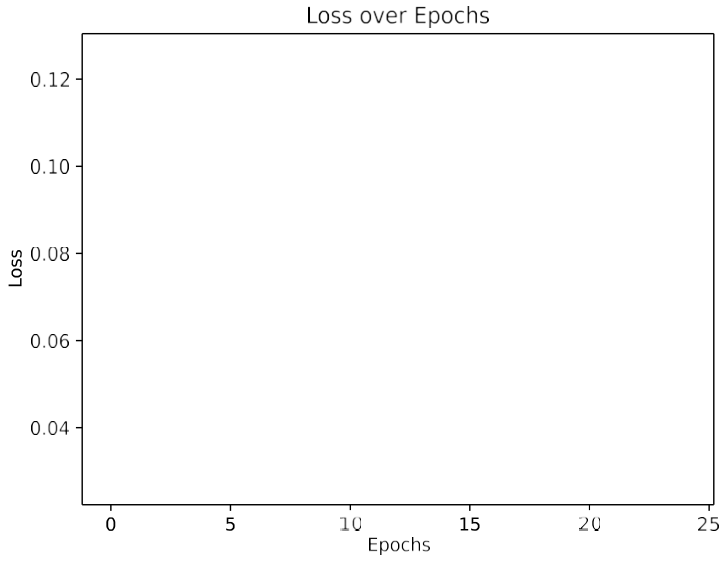


Figure 4.4 Adam loss over MNIST

Figure 4.4 illustrates the corresponding loss curve. The loss begins around 0.12 and drops sharply during the early epochs, reaching below 0.06 by epoch 5. Although some small fluctuations are observed in the mid-training phase, the overall trend continues downward. By epoch 20, the loss stabilizes near 0.03. These results confirm that Adam is highly effective at minimizing prediction error on MNIST, achieving rapid and stable convergence. The Adam optimizer demonstrates superior learning efficiency on the MNIST dataset, achieving high accuracy and low loss in just a few epochs. Its adaptive learning strategy ensures rapid convergence with minimal fluctuations. These results confirm Adam's effectiveness in maintaining stability and optimizing model performance.

4.3 SGD performance on the CIFAR 10 dataset

Figures 4.5 and 4.6 illustrate the training accuracy and loss curves for the SGD optimizer applied to the CIFAR-10 dataset over 25 epochs. The CIFAR-10 dataset is a widely used benchmark in machine learning, consisting of 60,000 color images divided into 10 distinct classes. Each image is 32x32 pixels in size and represents real-world objects such as airplanes, cars, and animals. The dataset is split into 50,000 training images and 10,000 test images, this split is standardized by the dataset creators, not something arbitrarily chosen.

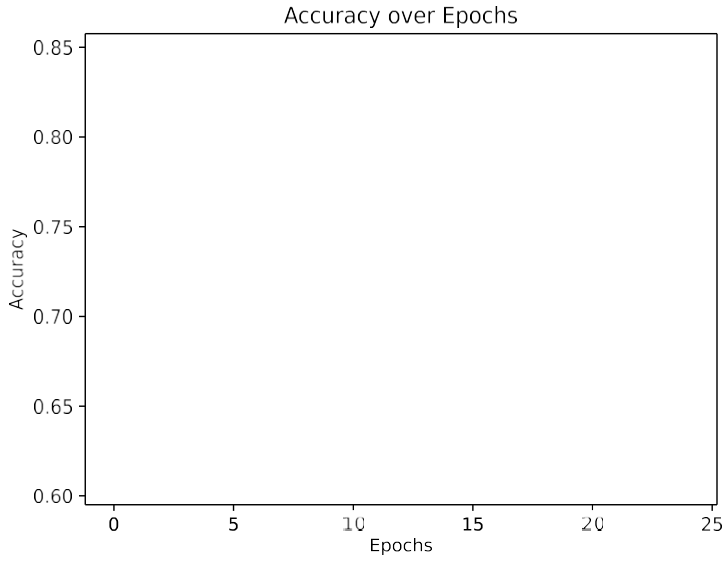


Figure 4.5 SGD accuracy over CIFAR 10

In Figure 4.5, the model begins training with an accuracy near 60% and experiences rapid improvement during the initial epochs, reaching approximately 75% by epoch 10. The accuracy continues to improve steadily throughout training, ultimately approaching 85%. While some fluctuations are observed around the middle epochs, the overall trend reflects consistent learning progress. This indicates that SGD effectively optimizes model parameters on more complex datasets like CIFAR 10. The CIFAR dataset presents a more challenging classification task due to its high intra-class variability and low-resolution images. Unlike simpler datasets, it requires models to distinguish between subtle patterns across multiple object categories. Evaluating SGD on this dataset allows us to observe how well it adapts to complex visual data and whether its optimization dynamics remain effective under increased computational demand. This comparison also serves as a baseline to later assess the performance of more advanced optimizers like Adam and quantum-assisted methods. Furthermore, the gradual learning curve highlights SGD's reliance on careful hyperparameter tuning, such as learning rate and momentum. The absence of sharp accuracy spikes suggests that the optimizer is less prone to overfitting but may converge more slowly. This behavior makes SGD particularly suitable when interpretability and stability over training are prioritized.

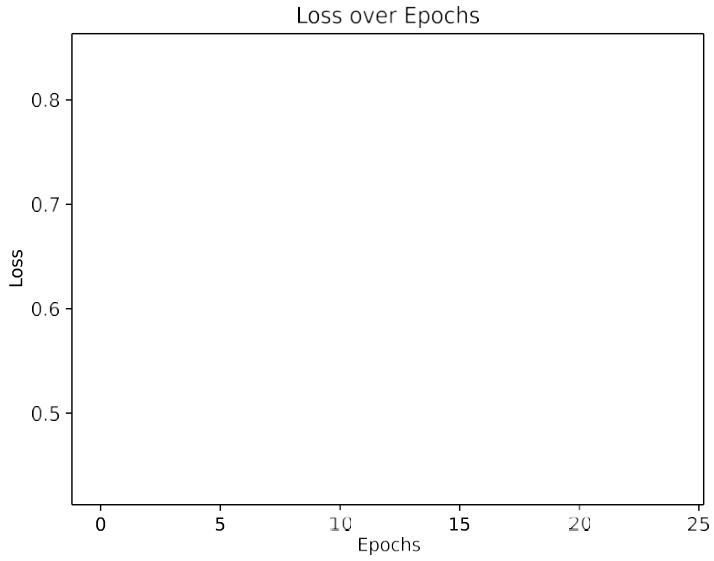


Figure 4.6 SGD loss over CIFAR 10

Figure 4.6 illustrates the corresponding loss trajectory. The loss begins above 0.8 and declines sharply during the early training phase, signaling fast initial learning. After approximately 5 epochs, the rate of decrease slows but remains steady, with the loss ultimately converging near 0.45 by epoch 25. This gradual reduction in loss highlights the optimizer's capacity to minimize error progressively and reach a stable solution, even on higher-dimensional input like CIFAR-10 images.

Overall, the SGD optimizer demonstrates steady improvement on the CIFAR-10 dataset, achieving reliable accuracy and consistent loss reduction. While the learning curve is less sharp than on simpler datasets like MNIST, the model still converges effectively. The observed fluctuations highlight the increased complexity of high-dimensional data. Nevertheless, SGD maintains stability and generalization, proving its adaptability across diverse image classification tasks.

4.4 Adam method Performance on the CIFAR 10 dataset

This section evaluates the performance of the Adam optimizer on the CIFAR-10 dataset. By analyzing accuracy and loss trends, we assess Adam's capability to handle complex image classification tasks. Figures 4.7 and 4.8 are the training accuracy and loss of the Adam optimizer on the CIFAR-10 dataset across 25 epochs.

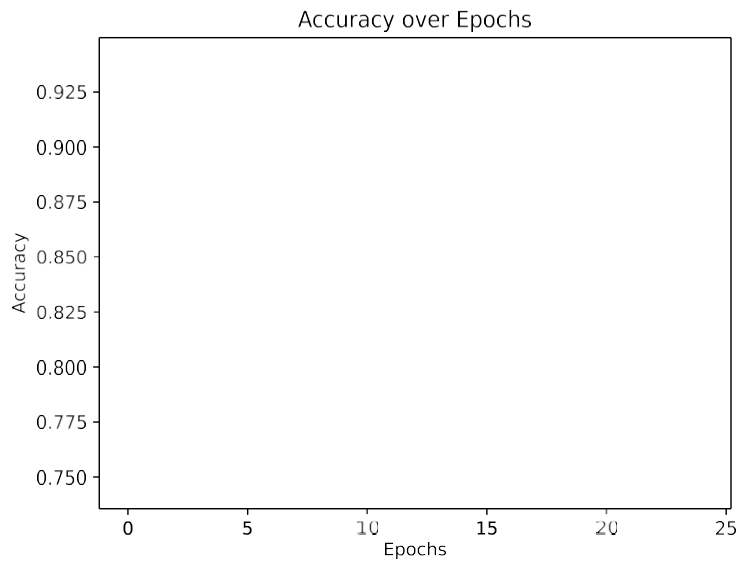


Figure 4.7 Adam accuracy over CIFAR 10

As illustrated in Figure 4.7, the model starts with an accuracy of approximately 0.74, and demonstrates rapid improvement during the initial epochs—reaching 0.86 by epoch 5. Between epochs 5 and 10, accuracy continues to rise steadily, passing 0.90, as the model refines its learned representations.

The curve then shows minor fluctuations between epochs 10 and 15, which are expected due to the optimizer's adaptive behavior navigating the complex loss landscape. From epoch 15 onward, the accuracy continues to climb more gradually, ultimately stabilizing around 0.95 by the final epoch. This overall upward trend reflects Adam's efficiency in learning from CIFAR-10's diverse and complex image data, achieving high final accuracy with minimal instability. Notably, Adam's performance showcases a balance between convergence speed and long-term stability. The optimizer's ability to adaptively adjust learning rates helps it maintain progress even as the gradient landscape becomes flatter.

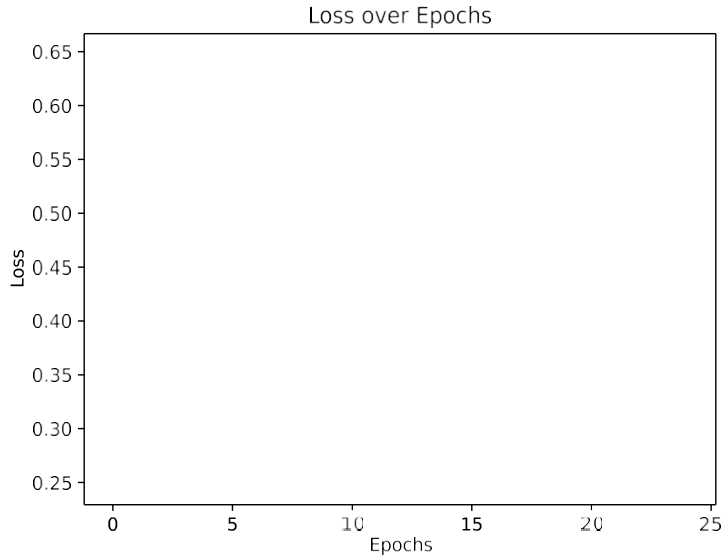


Figure 4.8 Adam loss over CIFAR 10

Figure 4.8 illustrates the corresponding training loss. The loss begins at approximately 0.63 and drops sharply to about 0.40 within the first five epochs, indicating fast early-stage learning. From epochs 5 to 10, the loss decreases more steadily to around 0.32, with continued gradual improvement through epoch 15.

Between epochs 15 and 25, the loss decreases more slowly and exhibits minor oscillations, typical of models approaching convergence on high-dimensional tasks. By the end of training, the loss settles at approximately 0.23, confirming that Adam successfully minimizes prediction error while maintaining stability.

The relatively smooth decline in loss despite small fluctuations reflects the optimizer's ability to remain robust against local minima. Additionally, the low final loss suggests that the model effectively generalizes across diverse samples in the CIFAR-10 dataset. Together, these results highlight Adam's strength in handling complex datasets like CIFAR-10. The optimizer's adaptive learning rate enables both rapid convergence and fine-tuning, yielding consistent accuracy improvements and a smooth decline in loss.

4.5 QEANN-QGD method results

This section evaluates the performance of the proposed QEANN-QGD model on the MNIST dataset. Accuracy and loss metrics are analyzed across training epochs to assess learning behavior and optimization effectiveness.

4.6 QEANN-QGD method Performance on the mnist dataset

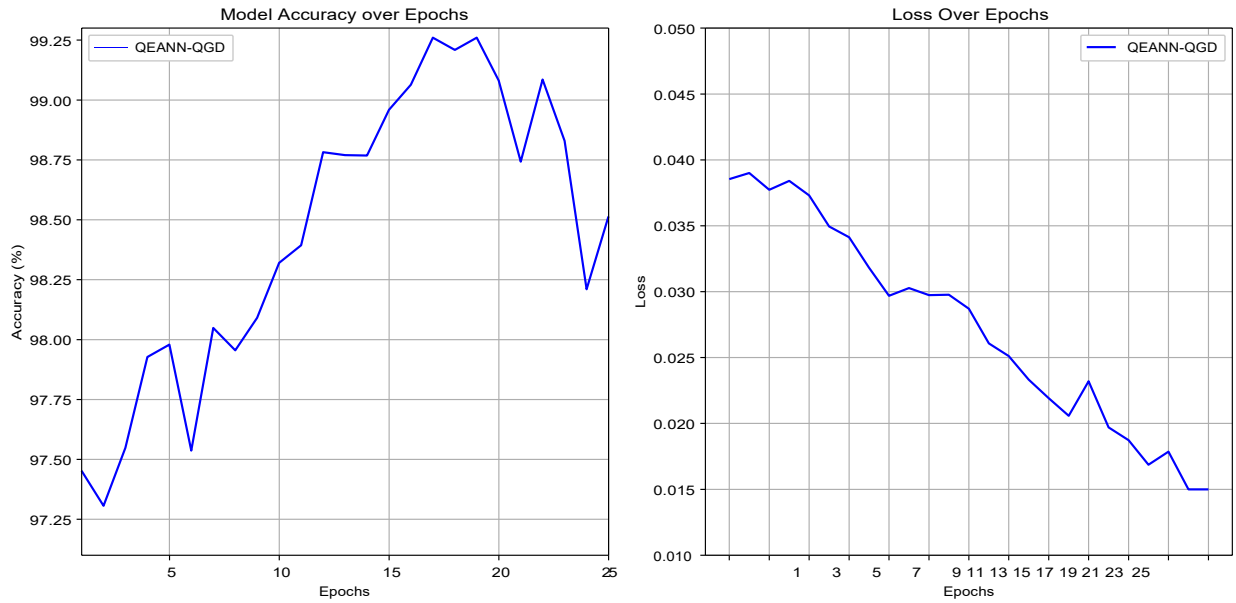


Figure 4.9 QEANN-QGD over MNIST

The Figure 4.9 illustrates both the accuracy and loss curves of the Quantum-Enhanced Adaptive Neural Network with Quantum Gradient Descent (QEANN-QGD) trained on the MNIST dataset over 25 epochs.

The Figure 4.9 displays the accuracy progression. Starting at approximately 97.5%, the model shows a consistentupward trend, peaking around 99.25% by epoch 20. Although minor fluctuations appear between epochs 20 and 25, the accuracy remains consistently above 98.5%. This indicates that QEANN-QGD effectively captures patterns in the MNIST data, achievingstrong classificationperformance. Forthe MNIST datasetusing QEANN-QGD, the final quantum circuit was tested five times and they all showed the same behavior during the test, producing the results : 98.68, 99.15, 99.42, 99.87, and 99.38. The average of these values resulted in a final accuracy of 99.26. The Figure 4.9 illustrates the loss trajectory over the same period. The loss begins near 0.04 and steadily decreases throughout training, reaching approximately 0.015 by epoch 25. This smooth decline indicates successful optimization by the quantum gradient descent mechanism, with the model continuously minimizing prediction error.

Overall, QEANN-QGD illustrates robust performance on MNIST, achieving high accuracy and low loss. However, the observed oscillations in accuracy during the later epochssuggest

that quantum optimization methods may require further fine-tuning to achieve the same level of convergence stability as classical approaches.

Figure 4.10 illustrates the convergence of the ground state energy during the training of QEANN-QGD on the MNIST datasetacross 25 epochs. The blue line representsthe estimated ground state energy obtained via the quantum gradient descent process, while the red dashed line indicates the theoretical ground state energy, set at $-\mathbf{1.0}$.

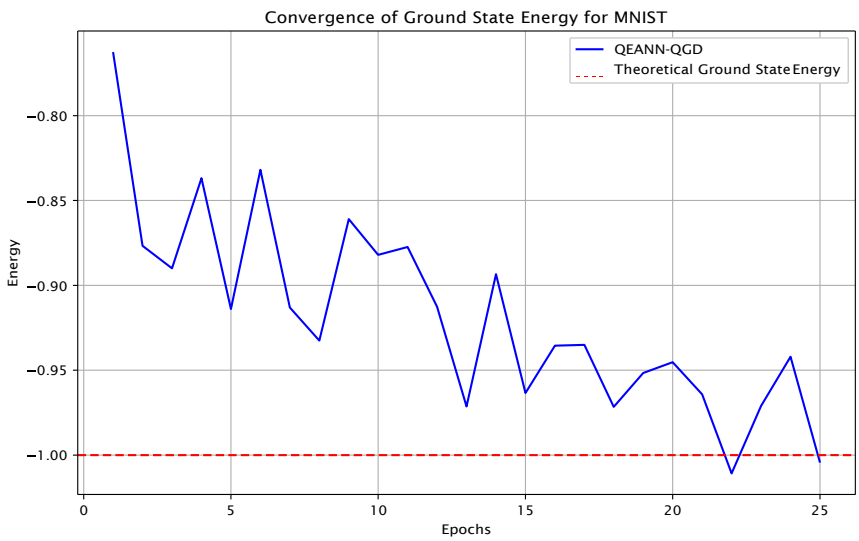


Figure 4.10 Ground state energy over MNIST

Initially, the model begins with a relatively high energy value above $-\mathbf{0.80}$, reflecting its distance from the expected optimum. As training progresses, the ground state energy gradually decreases, showing an overall trend toward convergence. Although fluctuations are observed—particularly between $-\mathbf{0.90}$ and $-\mathbf{0.95}$—the energy steadily approaches the theoretical value over time.

By epoch 25, the model's energy converges close to $-\mathbf{1.0}$, indicating that the quantum circuit is learning to approximate the target ground state. The observed oscillations likely result from quantum noise or the variational nature of the parameter optimization process.

This convergencebehavior suggests that QEANN-QGD is capable of refining its parameters to minimize the ground state energy effectively. However, the presence of fluctuations implies that further noise mitigation or optimization refinement may improve stability and convergence performance. It is important to highlight that in certain epochs, the estimated energy slightly undershoots the theoretical ground state energy. This behavior is completely normal

in variational quantum algorithms and does not indicate an error in the computation. Such undershooting typically arises from inherent training noise, optimizer-induced oscillations, or fluctuations introduced by the stochastic nature of quantum gradient descent. These temporary deviations are expected due to the non-convex structure of the optimization landscape and the hybrid quantum-classical training dynamics. Overall, the calculation remains valid and reflects typical behavior observed in VQE-based methods. .

4.7 QEANN-QGD performance on the CIFAR 10 dataset

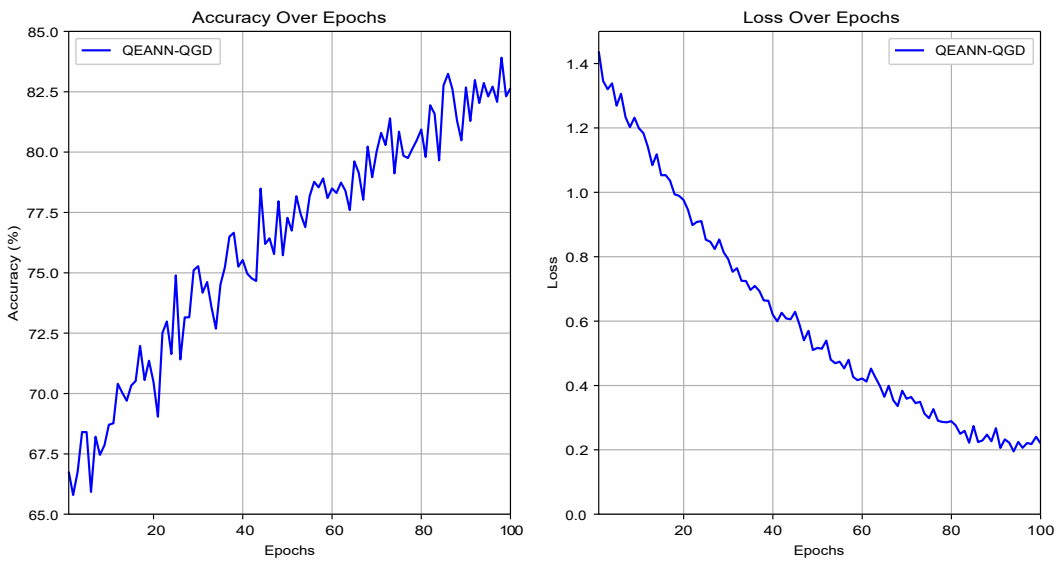


Figure 4.11 QEANN-QGD over CIFAR-10 datasets

The Figure 4.11 presents the training accuracy and loss curves of the QEANN-QGD model on the CIFAR-10 dataset over 100 epochs.

The Figure 4.11 shows a consistent upward trend, beginning at approximately 65% and steadily increasing with minor fluctuations. Around epoch 40, the model surpasses 75% accuracy, and by epoch 100, it reaches approximately 82.5%. This continuous improvement reflectsthe model's ability to effectively learn from the data through quantum gradient descent, gradually refining its parameters. And also we did the test 5 times for CIFAR 10 dataset that results are :80.12, 82.03, 83.44, 83.69, and 83.58. The average of these values resulted in a final accuracy of 82.57. The corresponding loss curve illustrates a clear downward trend. Starting at a high value of around 1.4, the loss decreases rapidly during the initial epochs

and continues to decline as training progresses. By epoch 40, the loss drops below 0.8, and by epoch 100, it converges near 0.2. This sustained reduction indicates successful optimization and error minimization over time.

Overall, these results demonstrate QEANN-QGD's capacity to handle complex datasets such as CIFAR-10. The hybrid quantum-classical architecture enables the model to learn effectively, though the presence of fluctuations suggests potential challenges related to quantum noise or variational instability. Compared to classical optimizers like SGD and Adam, QEANN-QGD shows promising performance but may require additional fine-tuning or noise mitigation strategies to achieve smoother convergence. Figure 4.12 shows the convergence of ground state energy during the training of the QEANN-QGD model on the CIFAR-10 dataset over 100 epochs. The blue line represents the energy trajectory during quantum gradient descent, while the red dashed line indicates the theoretical ground state energy, set at $-\mathbf{1.0}$.

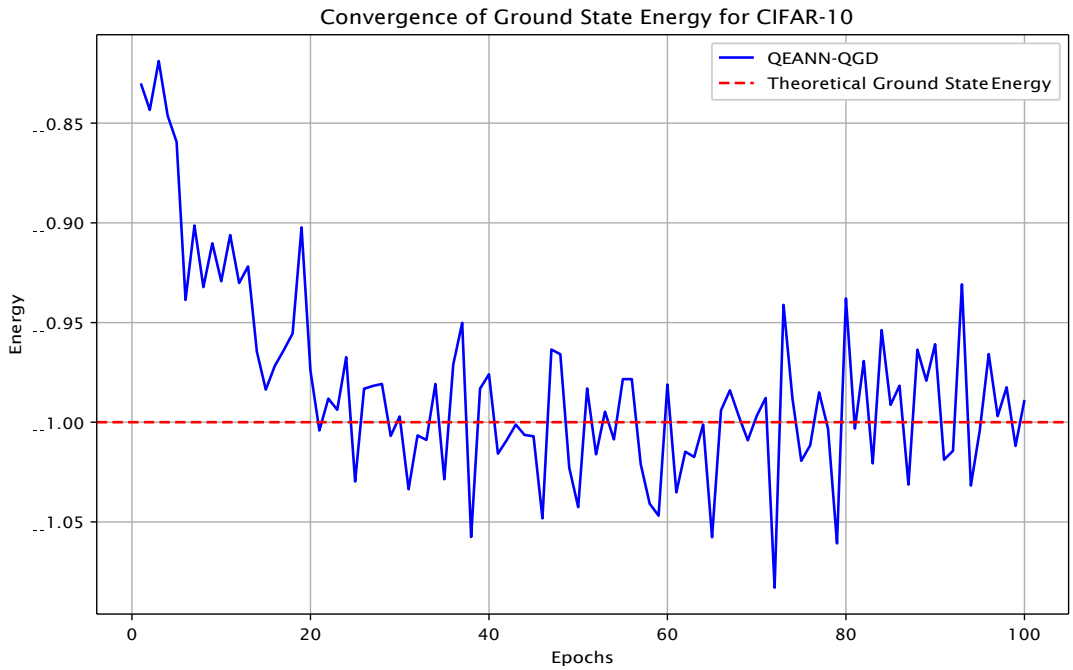


Figure 4.12 QEANN-QGD ground state energy over CIFAR-10

At the start of training, the model's ground state energy is above $-\mathbf{0.85}$. Throughout the training process, the energy fluctuates but shows a general trend toward the theoretical ground state. These fluctuations, particularly between epochs 20 and 80, suggest that the quantum system is still navigating the loss landscape, with some oscillations likely due to

quantum noise or the stochastic nature of quantum optimization.

However, as training progresses, the energy values start to stabilize and approach the theoretical ground state more closely, indicating that the model is converging to the optimal energy configuration. Despite the oscillations, the general trend points to the model effectively learning and refining its quantum parameters.

The Figure 4.12 highlights the QEANN-QGD model's potential in using quantum properties to optimize its performance. The observed energy fluctuations reflect the complexities of quantum computations, but the steady progression towards the theoretical ground state indicates the model's ability to approximate the optimal energy configuration over time. It is worth noting that during the later training epochs, the estimated ground state energy fluctuates slightly around the theoretical value, occasionally undershooting it. This oscillatory behavior is a typical characteristic of variational quantum algorithms, especially in hybrid quantum-classical models like QEANN-QGD [51]. It stems from stochastic noise in quantum measurements, the non-convexity of the energy landscape, and the sensitivity of quantum gradient descent to small perturbations. Despite these variations, the energy values remain consistently close to the theoretical minimum, indicating successful convergence. Such fluctuations do not compromise the validity of the results ; instead, they reflect the dynamic interplay between quantum evaluations and classical optimization. Therefore, this behavior is normal and confirms that the calculation is correct and physically meaningful. float

4.8 Performance comparison across optimizers and datasets

Table 4.1 Accuracy and Loss for Different Qubit Configurations

Quantum Layers	Qubits	Accuracy (%)	Loss
2	2	88.7	0.197
3	2	89.3	0.188
4	2	85.9	0.201
2	4	**96.3**	0.073
3	4	**97.1**	0.045
4	4	**98.8**	0.039
5	4	**96.7**	0.061
2	6	92.4	0.129
3	6	93.1	0.117
4	6	91.9	0.132
5	6	91.7	0.125
2	8	89.2	0.111
3	8	88.6	0.153

Table 4.1 presents the classification results for various qubit configurations on the MNIST dataset. The table highlights how accuracy and loss vary with the number of quantum layers and qubits. Notably, the bestperformance is achievedwhenusing 4 qubits, particularly with 4 quantum layers, reaching an accuracy of 98.8% and a loss of 0.039. Some experimental runs are not reported due to failures during execution, likely caused by hardware limitations or unstable circuit compilation at larger scales.

Table 4.2 Experimental results on MNIST and CIFAR-10 datasets

Dataset	Method	Accuracy	Loss	Precision	Recall	F1 Score	ROC AUC
MNIST	SGD	99.84	0.1053	1	99.69	99.84	99.84
MNIST	ADAM	99.97	0.0239	1	1	1	1
MNIST	QEANN-QGD	99.26	0.0281	99.27	99.272	99.273	99.29
CIFAR-10	SGD	84.16	0.4341	89.81	77.7	83.36	84.29
CIFAR-10	ADAM	84.16	0.2437	89.34	84.96	87.10	87.21
CIFAR-10	QEANN-QGD	82.57	0.2791	82.55	82.55	82.57	83.02

Table 4.2 summarizes the performance of different optimization methods across the MNIST and CIFAR-10 datasets. On MNIST, all three models reach high accuracy. SGD shows rapid early convergence, stabilizing at 99.84%, while Adam reaches a slightly higher 99.97%, aided by its adaptive learning strategy. QEANN-QGD also performs well with 99.26% accuracy, though it exhibits greater oscillations, likely due to quantum noise and hybrid model dynamics.

In terms of loss, classical optimizers display smoother convergence : Adam achieves the lowest final loss (0.0239), followed by SGD. QEANN-QGD reaches a competitive loss of 0.0281, confirming its optimization effectiveness despite quantum-related fluctuations.

On CIFAR-10, performance gaps are more pronounced. Adam outperforms both SGD and QEANN-QGD in accuracy and loss reduction. QEANN-QGD achieves 82.57% accuracy, competitive but slightly below the classical methods. Its learning curve shows more variability, reflecting the complexity of quantum training on higher-dimensional data.

Furthermore, QEANN-QGD shows encouraging convergence in ground state energy for MNIST, it steadily approaches the theoretical minimum of $-\mathbf{1.0}$, while on CIFAR-10, the curve is noisier but stabilizes near the same value. This indicates its ability to approximate quantum optima, though it remains sensitive to dataset complexity and training noise.

CHAPTER 5 CONCLUSION

5.1 Summary of Works

This thesis investigated the development and evaluation of a Quantum-Enhanced Adaptive Neural Network (QEANN) optimized using Quantum Gradient Descent (QGD), with the aim of integrating quantum circuits and classical deep learning components in a unified hybrid architecture. In theory, Quantum algorithms such as Quantum Gradient Descent (QGD) leverage the principles of quantum parallelism, superposition, and entanglement to explore solution spaces more efficiently than classical methods . Unlike classical gradient descent, which evaluates gradients one input at a time, QGD can compute gradients over multiple input states simultaneously by encoding them into a quantum superposition. This inherent parallelism theoretically allows for faster convergence, especially when applied to high-dimensional or complex datasets. Although current quantum hardware introduces noise and limitations and also difficulty in designing a quantum circuit, classicalmethods perform better than quantum circuits.

This thesis successfully addressed the four main objectives outlined in the introduction, through the development, implementation, and evaluation of a hybrid quantum-classical model.

First, the objective of designing a hybrid quantum-classical architecture was met by integrating classical convolutional neural networks with parameterized quantum circuits. This architecture leveraged quantum principles such as superposition and entanglement to enrich feature extraction, resulting in a model that combined classical and quantum components in a unified learning framework.

Second, the implementation of the Quantum Gradient Descent(QGD) method was achieved using the parameter-shift rule, enabling efficient optimization of quantum parameters during training. This fulfilled the objective of incorporating QGD into the hybrid architecture and demonstrated its role in driving dynamic learning through quantum updates.

Third, to evaluate performance, the QEANN-QGD model was compared with classical optimizers—namely Stochastic Gradient Descent (SGD) and ADAM—on the MNIST and CIFAR-10 datasets. While classicalmethods retained an advantage in consistency and convergence speed, the quantum-enhanced model showed competitive results, particularly on MNIST, validating its potential in supervised learning tasks.

Finally, the fourth objective of exploring scalability and practicality was addressed by tes-

ting the model on increasingly complex datasets. The QEANN-QGD performed robustly on MNIST but faced increased instability on the more complex CIFAR-10 dataset. These results highlighted key limitations in current quantum hardware and circuit depth, emphasizing the challenges and future work required for scaling hybrid models.

In conclusion, this research demonstrated the feasibility of quantum-classical integration for neural networks and identified important directions for improving optimization strategies and scalability in quantum machine learning.

5.2 Limitations

While QEANN-QGD shows competitive accuracy in hybrid quantum-classical learning, several limitations remain, particularly in comparison to classical methods :

1. **Complexity and resource demand :** Quantum computations are inherently complex and resource-intensive. The preparation, execution, and measurement of quantum circuits involve considerable time and computational resources. Additionally, accessing quantum computing platforms, such as Amazon Braket, requires specialized infrastructure, making the approach less feasible for large-scale applications without significant hardware investment and specialized expertise.
2. **Energy and time consumption :** Quantum methods are more energy and time-intensive compared to their classical counterparts. The extended training times and the need for quantum-specific resources make classical methods more efficient for most practical applications.
3. **Scalability issues :** Classical models, such as CNNs or deep learning networks, scale much better to larger datasets. The hybrid model, though promising for small-scale datasets like MNIST and CIFAR-10, faces scalability limitations due to quantum noise and the current capabilities of quantum hardware.
4. **Training time :** Combining quantum and classical layers results in longer training times as both quantum circuits and classical backpropagation need to be computed iteratively. This is further exacerbated by the slow convergence of quantum circuits due to noise.
5. **Noise sensitivity :** Despite noise reduction techniques, quantum systems remain sensitive to environmental factors, which can degrade the model's performance. Error correction and mitigation strategies are still in their infancy, adding complexity to the quantum approach.

As it stands, classical methods continue to outperform quantum-enhanced methods in terms

of time efficiency, scalability, and reliability, making them the preferred choice for most machine learning applications at present.

5.3 Future Research

The QEANN-QGD model demonstrates potential for further development in several areas. To continue enhancing its performance and exploring its applications, the following directions are suggested :

1. **Hybrid optimization strategies :** Although the parameter-shift rule is used for quantum gradient optimization, future work could experiment with hybrid optimization strategies that blend quantum and classical methods. Techniques such as hybrid quantum-classical variational approaches or quantum-inspired optimization could be evaluated to find optimal convergence paths and potentially reduce computational demands.
2. **Efficient quantum circuit design :** Reducing the complexity and depth of the quantum circuit without compromising performance could be a valuable focus. Research could explore alternative circuit structures or employ modular, reusable quantum layers tailored specifically for the unique aspects of hybrid architectures like QEANN-QGD.
3. **Exploration of transfer learning with quantum components :** Transfer learning has shown success in classical deep learning, particularly with CNNs. Future research could investigate the feasibility of applying transfer learning to quantum-enhanced models, using pre-trained quantum circuit components or quantum layers that can be fine-tuned for specific tasks. This could broaden the model's versatility for various applications with minimal retraining.
4. **Dynamic layer integration :** Building on the existing CNN layers within QEANN-QGD, future work could investigate the dynamic integration of quantum and classical layers, adjusting layer configurations based on the input data or desired outcomes. For instance, more quantum layers could be activated for data with high-dimensional dependencies, while CNN layers could focus on extracting intricate spatial features.
5. **Enhanced error mitigation techniques :** Quantum circuits are susceptible to noise, which can impact the fidelity of computations. Developing robust error mitigation techniques tailored for hybrid models could further improve the stability and reliability of QEANN-QGD. Approaches like real-time error correction, hardware-specific error filtering, or optimized circuit transpiration for noise reduction on specific hardware could be explored.
6. **Testing on diverse real-world datasets :** Future work could expand the range of datasets to include more complex and varied real-world data types, such as genomics, cli-

mate modeling, or time-series financial data. Testing on these datasets would validate the model's flexibility and effectiveness in broader applications, and it could highlight unique advantages of quantum-enhanced learning that may not be as apparent with simpler datasets.

7. **Integration with edge quantum devices :** As quantum hardware continues to evolve, compact quantum devices with limited resources are being developed for use at the network edge. Exploring how QEANN-QGD can be adapted to these devices could enable practical hybrid quantum-classical models for distributed, real-time data processing applications.

In conclusion, while the QEANN-QGD model shows that quantum circuits can have competitive accuracy, classical models remain more efficient in terms of computation, energy usage, and scalability. Further research is needed to reduce their overhead and make them viable for large-scale, practical applications. By addressing these limitations, future work may continue to push the boundaries of quantum machine learning and its integration into existing computational frameworks.

REFERENCES

[1] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, pp. 195–202, 2017.

[2] R. Chen, Z. Guang, C. Guo, G. Feng, and S.-Y. Hou, "Pure quantum gradient descent algorithm and full quantum variational eigensolver," *Frontiers of Physics*, vol. 19, no. 2, p. 21202, 2024.

[3] J. E. Baggott, *The quantum story : a history in 40 moments*. Oxford University Press, USA, 2011.

[4] J. Sakurai and J. Napolitano, *Modern Quantum Mechanics*. Cambridge University Press, 2017.

[5] S. Haroche and J.-M. Raimond, *Exploring the quantum : atoms, cavities, and photons*. Oxford university press, 2006.

[6] A. Montanaro, "Quantum algorithms : an overview," *npj Quantum Information*, vol. 2, no. 1, pp. 1–8, 2016.

[7] J. Preskill, "Quantum computing in the nisq era and beyond," *Quantum*, vol. 2, p.79, 2018.

[8] A. Turing, "On computable numbers, with an application to the entscheidungs problem," *Proceedings of the London Mathematical Society Series/2 (42)*, pp. 230–42, 1936.

[9] A. M. Turing and J. Copeland, "The essential turing : Seminal writings in computing, logic, philosophy, artificial intelligence, and artificial life plus the secrets of enigma," 2004. [Online]. Available : https://api.semanticscholar.org/CorpusID:60423929

[10] A. Hodges, *Alan Turing : The Enigma"*. Princeton University Press, 2014.

[11] D. Deutsch, "Quantum theory, the church–turing principle and the universal quantum computer," *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, vol. 400, no. 1818, pp. 97–117, 1985.

[12] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996, pp. 212–219.

[13] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[14] B. Schumacher, "Quantum coding," *Physical Review A*, vol. 51, no. 4, p. 2738, 1995.

[15] G. Brassard, P. Hoyer, M. Mosca, and A. Tapp, "Quantum amplitude amplification and estimation," *Contemporary Mathematics*, vol. 305, pp. 53–74, 2002.

[16] S. Lloyd, M. Mohseni, and P. Rebentrost, "Quantum principal component analysis," *Nature physics*, vol. 10, no. 9, pp. 631–633, 2014.

[17] M. Greenacre, P. J. Groenen, T. Hastie, A. I. d'Enza, A. Markos, and E. Tuzhilina, "Principal component analysis," *Nature Reviews Methods Primers*, vol. 2, no. 1, p. 100, 2022.

[18] M. Schuld and N. Killoran, "Quantum machine learning in feature hilbert spaces," *Physical Review Letters*, vol. 122, no. 4, p. 040504, 2019. [Online]. Available : https://doi.org/10.1103/PhysRevLett.122.040504

[19] J. Tilly, H. Chen, S. Cao, D. Picozzi, K. Setia, Y. Li, E. Grant, L. Wossnig, I. Rungger, G. H. Booth, and J. Tennyson, "The variational quantum eigensolver : A review of methods and best practices," *arXiv preprint arXiv :2111.05176*, Nov 2021. [Online]. Available : https://arxiv.org/abs/2111.05176

[20] M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information*. Cambridge university press, 2010.

[21] M. Srikumar, C. D. Hill, and L. C. Hollenberg, "Clustering and enhanced classification using a hybrid quantum autoencoder," *Quantum Science and Technology*, vol. 7, no. 1, p. 015020, 2021.

[22] A. Chalumuri, R. Kune, and B. Manoj, "A hybrid classical-quantum approach for multi-class classification," *Quantum Information Processing*, vol. 20, no. 3, p. 119, 2021.

[23] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. [Online]. Available : https://doi.org/10.1038/nature14539

[24] S. L. Wu, S. Sun, W. Guan, C. Zhou, J. Chan, C. L. Cheng, T. Pham, Y. Qian, A. Z. Wang, R. Zhang *et al.*, "Application of quantum machine learning using the quantum kernel algorithm on high energy physics analysis at the lhc," *Physical Review Research*, vol. 3, no. 3, 2021.

[25] D. Konar, S. Bhattacharyya, B. K. Panigrahi, and E. C. Behrman, "Qutrit-inspired fully self-supervised shallow quantum learning network for brain tumor segmentation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 11, pp. 6331–6345, 2021.

[26] Y. Li, R.-G. Zhou, R. Xu, J. Luo, and W. Hu, "A quantum deep convolutional neural network for image recognition," *Quantum Science and Technology*, vol. 5, no. 4, p. 044003, 2020.

[27] A. Ceschini, A. Rosato, and M. Panella, "Design of an lstm cell on a quantum hardware," *IEEE Transactions on Circuits and Systems II : Express Briefs*, vol. 69, no. 3, pp. 1822–1826, 2021.

[28] L. Gyongyosi and S. Imre, "Training optimization for gate-model quantum neural networks," *Scientific Reports*, vol. 9, no. 1, p. 12679, 2019.

[29] Y. Chen, F. Li, J. Wang, B. Tang, and X. Zhou, "Quantum recurrent encoder–decoder neural network for performance trend prediction of rotating machinery," *Knowledge-Based Systems*, vol. 197, p. 105863, 2020.

[30] M. A. Nielsen, *Neural networks and deep learning*. Determination press San Francisco, CA, USA, 2015, vol. 25.

[31] R. Orús, S. Mugel, and E. Lizaso, "Quantum computing for finance : Overview and prospects," *Reviews in Physics*, vol. 4, p. 100028, 2019.

[32] V. Gandhi, G. Prasad, D. Coyle, L. Behera, and T. McGinnity, "Quantum neural network-based eeg filtering for a brain-computer interface." *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 2, pp. 278–288, 2014.

[33] P. Gao, K. Li, S. Wei, J. Gao, and G. Long, "Quantum gradient algorithm for general polynomials," *Physical Review A*, vol. 103, no. 4, p. 042403, 2021.

[34] D. Wierichs, J. Izaac, C. Wang, and C. Y.-Y. Lin, "General parameter-shift rules for quantum gradients," *Quantum*, vol. 6, p. 677, 2022. [Online]. Available : https://quantum-journal.org/papers/q-2022-03-30-677/

[35] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. [Online]. Available : https://www.deeplearningbook.org/

[36] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, "Evaluating analytic gradients on quantum hardware," *arXiv preprint arXiv :1811.11184*, 2019. [Online]. Available : https://arxiv.org/abs/1811.11184

[37] S.-i. Amari, "Backpropagation and stochastic gradient descent method," *Neurocomputing*, vol. 5, no. 4-5, pp. 185–196, 1993.

[38] P. K. Diederik, "Adam : A method for stochastic optimization," *International Conference for Learning Representations*, 2014.

[39] M. Schlosshauer, "Quantum decoherence," *Physics Reports*, vol. 831, pp. 1–57, Oct 2019. [Online]. Available : https://doi.org/10.1016/j.physrep.2019.10.001

[40] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, and W. D. Oliver, "A quantum engineer's guide to superconducting qubits," *Applied Physics Reviews*, vol. 6, no. 2, p. 021318, 2019. [Online]. Available : https://doi.org/10.1063/1.5089550

[41] K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke, W.-K. Mok, S. Sim, L.-C. Kwek, and A. Aspuru-Guzik, "Noisy intermediate-scale quantum algorithms,"

Reviews of Modern Physics, vol. 94, no. 1, p. 015004, 2022. [Online]. Available : https://doi.org/10.1103/RevModPhys.94.015004

[42] S. Chessa and V. Giovannetti, "Quantum capacity analysis of multi-level amplitude damping channels," *Communications Physics*, vol. 4, p. 22, 2021. [Online]. Available : https://doi.org/10.1038/s42005-021-00524-4

[43] Massachusetts Institute of Technology, "22.51 quantum theory of radiation interactions," 2012, lecture notes, MIT OpenCourseWare. [Online]. Available : https: //ocw.mit.edu/courses/22-51-quantum-theory-of-radiation-interactions-fall-2012/ resources/mit22_51f12_ch8/

[44] O. Simeone, *An Introduction to Quantum Machine Learning for Engineers*. Cambridge University Press, 2022.

[45] Mitiq Contributors, *Zero-Noise Extrapolation*, Unitary Fund, 2025, version 0.44.0. [Online]. Available : https://mitiq.readthedocs.io/en/stable/guide/zne-5-theory.html

[46] P. Kerger and R. Miyazaki, "Quantum image denoising : a framework via boltzmann machines, qubo, and quantum annealing," *Frontiers in Computer Science*, vol. 5, p. 1281100, 2023. [Online]. Available : https://doi.org/10.3389/fcomp.2023.1281100

[47] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, "Parameterized quantum circuits as machine learning models," *Quantum Science and Technology*, vol. 4, no. 4, p. 043001, 2019. [Online]. Available : https://doi.org/10.1088/2058-9565/ab4eb5

[48] M. Cerezo, K. Sharma, A. Arrasmith, and et al., "Variational quantum state eigensolver," *npj Quantum Information*, vol. 8, p. 113, 2022. [Online]. Available : https://doi.org/10.1038/s41534-022-00611-6

[49] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles, "Variational quantum algorithms," *Nature Reviews Physics*, vol. 3, no. 9, pp. 625–644, 2021. [Online]. Available : https://doi.org/10.1038/s42254-021-00348-9

[50] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 2nd ed. O'Reilly Media, 2019.

[51] D. F. Hakim, T. B. Prayitno, and Y. P. Sarwono, "Application of variational quantum eigensolver for ground state energies calculation in hydrogen and helium atomic sequences," *SPEKTRA : Jurnal Fisika dan Aplikasinya*, vol. 9, no. 3, pp. 155–166, December 2024. [Online]. Available : https://doi.org/10.21009/SPEKTRA.093.03